US006502194B1

(12) **United States Patent**
Berman et al.

(10) **Patent No.: US 6,502,194 B1**
(45) **Date of Patent: Dec. 31, 2002**

(54) **SYSTEM FOR PLAYBACK OF NETWORK AUDIO MATERIAL ON DEMAND**

(75) Inventors: **Russell Todd Berman**, San Jose, CA (US); **Michael Andrew Radford**, Los Angeles, CA (US); **Brett Austin Kennedy**, Santa Monica, CA (US); **David Kiyoshi Matsumoto**, San Jose, CA (US)

(73) Assignee: **Synetix Technologies**, San Francisco, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/293,252**

(22) Filed: **Apr. 16, 1999**

(51) **Int. Cl.**[7] **H04L 12/00**; G06F 13/372

(52) **U.S. Cl.** **713/201**; 713/200; 709/231; 370/231; 370/352; 370/310; 370/468; 705/51; 705/26

(58) **Field of Search** 705/51, 57, 26, 705/27; 713/200, 201; 370/69.1, 231, 352, 354, 310, 468; 709/231, 233, 235

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,788,675 A		11/1988	Jones et al.	370/69.1
4,829,372 A		5/1989	McCalley et al.	358/86
5,544,228 A		8/1996	Wagner et al.	379/67
5,553,140 A		9/1996	Kubota et al.	380/10
5,635,979 A		6/1997	Kostreski et al.	348/13
5,790,423 A		8/1998	Lau et al.	364/514
5,822,537 A	*	10/1998	Katseff et al.	370/231
6,138,147 A	*	10/2000	Weaver et al.	707/104.1
6,222,838 B1	*	4/2001	Sparks et al.	370/352
6,246,672 B1	*	6/2001	Lumelsky	370/310

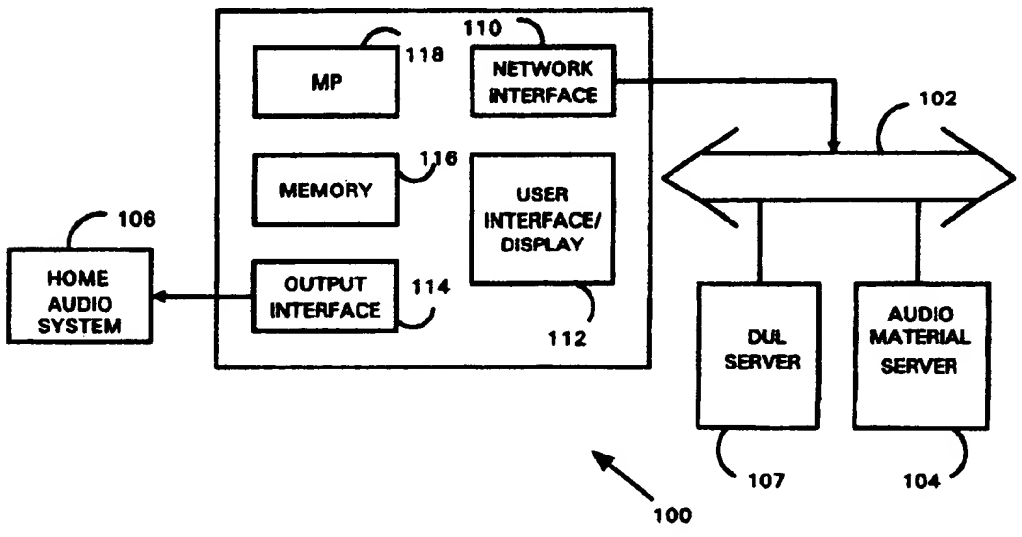
* cited by examiner

Primary Examiner—Ly V. Hua
(74) *Attorney, Agent, or Firm*—David A. Hall

(57) **ABSTRACT**

A playback unit resembling a home audio component, retrieves audio data from a remote server and plays them back in real time, using a home audio system, in response to user selection. The playback unit provides an interface between a network source for audio material, such as the Internet, and a conventional home audio system for playback. The playback unit has a relatively simple operating system that does not require a lengthy boot-up sequence, cannot be accessed by the user, and does not require the launch of special software to initiate playback. Access to audio material and distribution rights can be controlled by network servers. In this to way, the playback unit can retrieve audio material from the network on demand, thereby vastly expanding the range of music available for playback, and can reproduce that music using the home audio system for high quality playback in a comfortable setting, with controlled access to audio material and controlled distribution and duplication of the material.

24 Claims, 10 Drawing Sheets



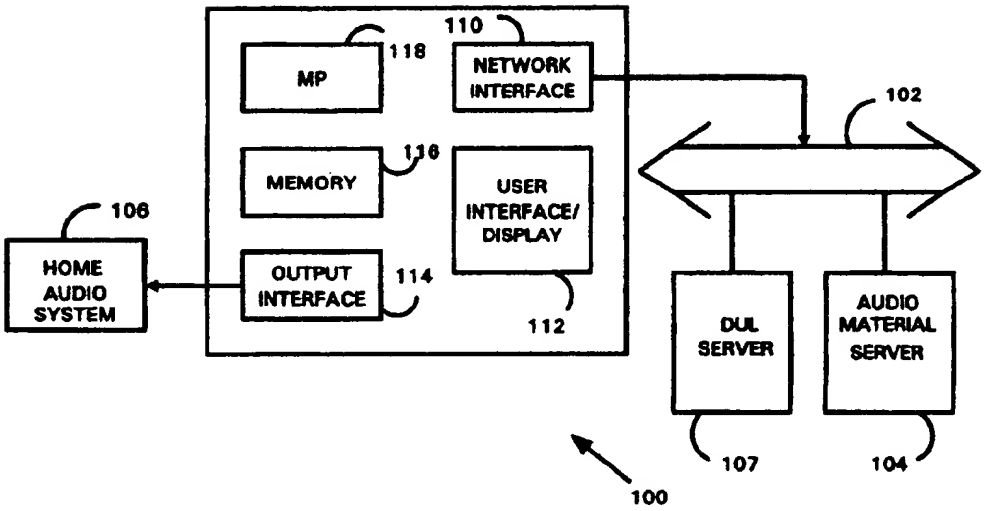

110
MP
118
NETWORK INTERFACE
102
MEMORY
116
USER INTERFACE/ DISPLAY
106
HOME AUDIO SYSTEM
OUTPUT INTERFACE
114
112
DUL SERVER
AUDIO MATERIAL SERVER
107
104
100

FIG. 1

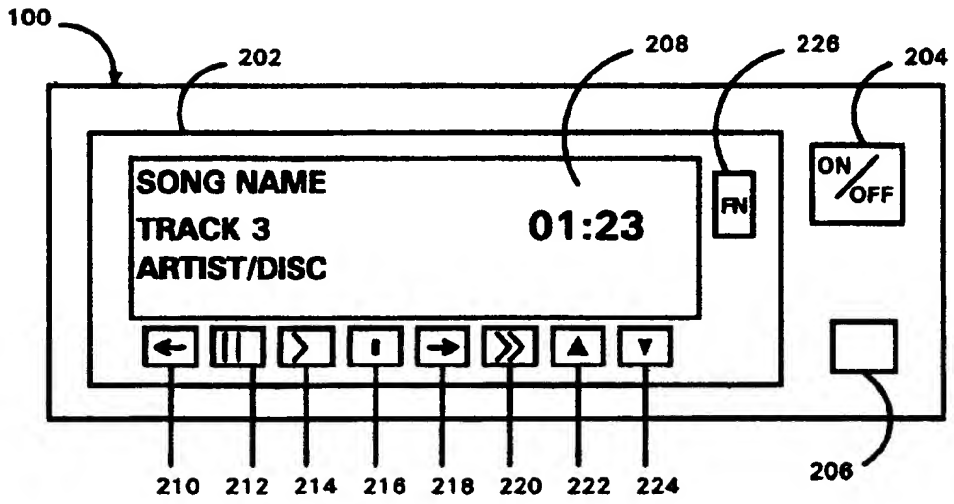

100
202
208
226
204
SONG NAME
TRACK 3
01:23
ARTIST/DISC
FN
ON/OFF
210
212
214
216
218
220
222
224
206

FIG. 2

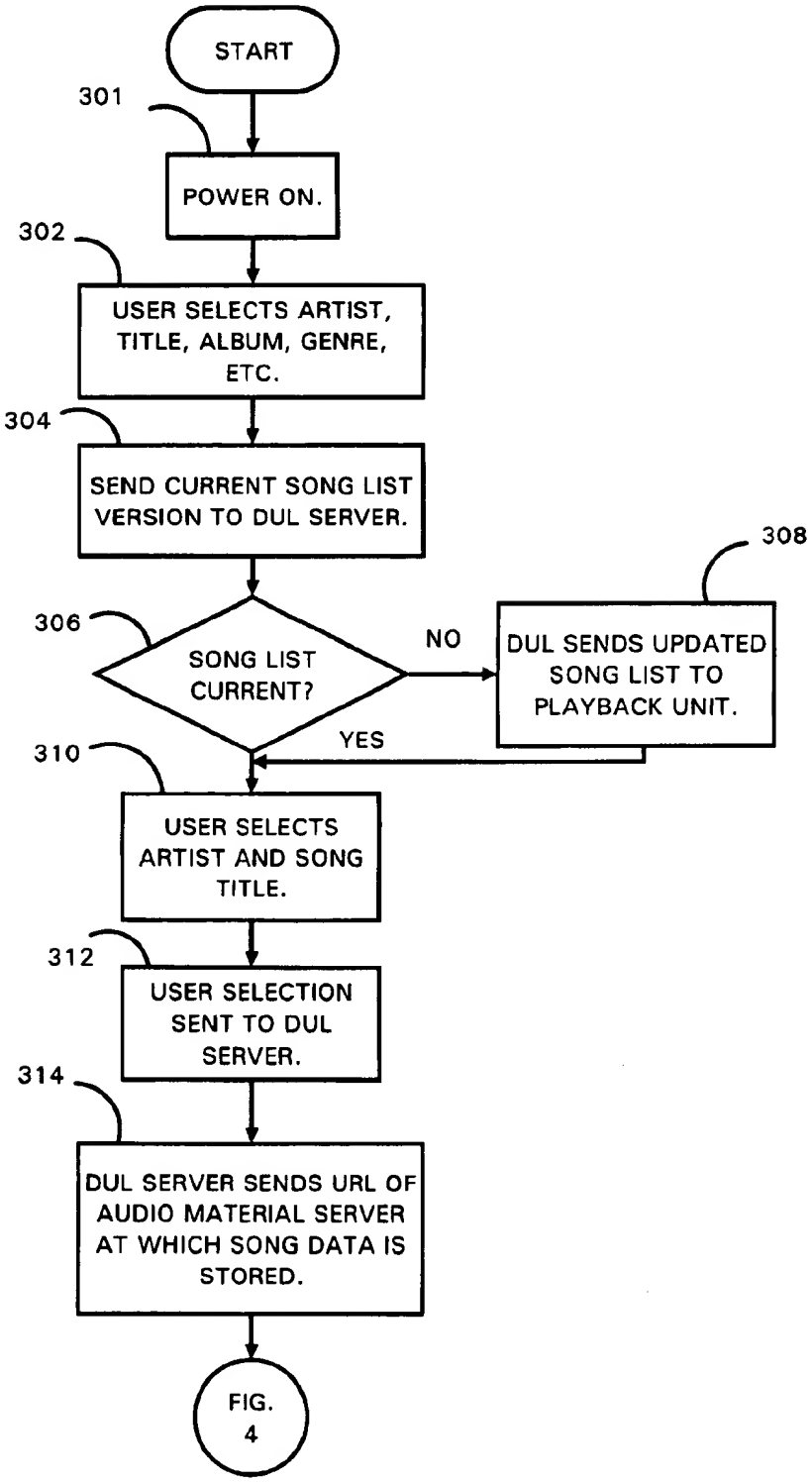

START
301
POWER ON.
302
USER SELECTS ARTIST, TITLE, ALBUM, GENRE, ETC.
304
SEND CURRENT SONG LIST VERSION TO DUL SERVER.
306
SONG LIST CURRENT?
NO
308
DUL SENDS UPDATED SONG LIST TO PLAYBACK UNIT.
YES
310
USER SELECTS ARTIST AND SONG TITLE.
312
USER SELECTION SENT TO DUL SERVER.
314
DUL SERVER SENDS URL OF AUDIO MATERIAL SERVER AT WHICH SONG DATA IS STORED.
FIG. 4

FIG. 3

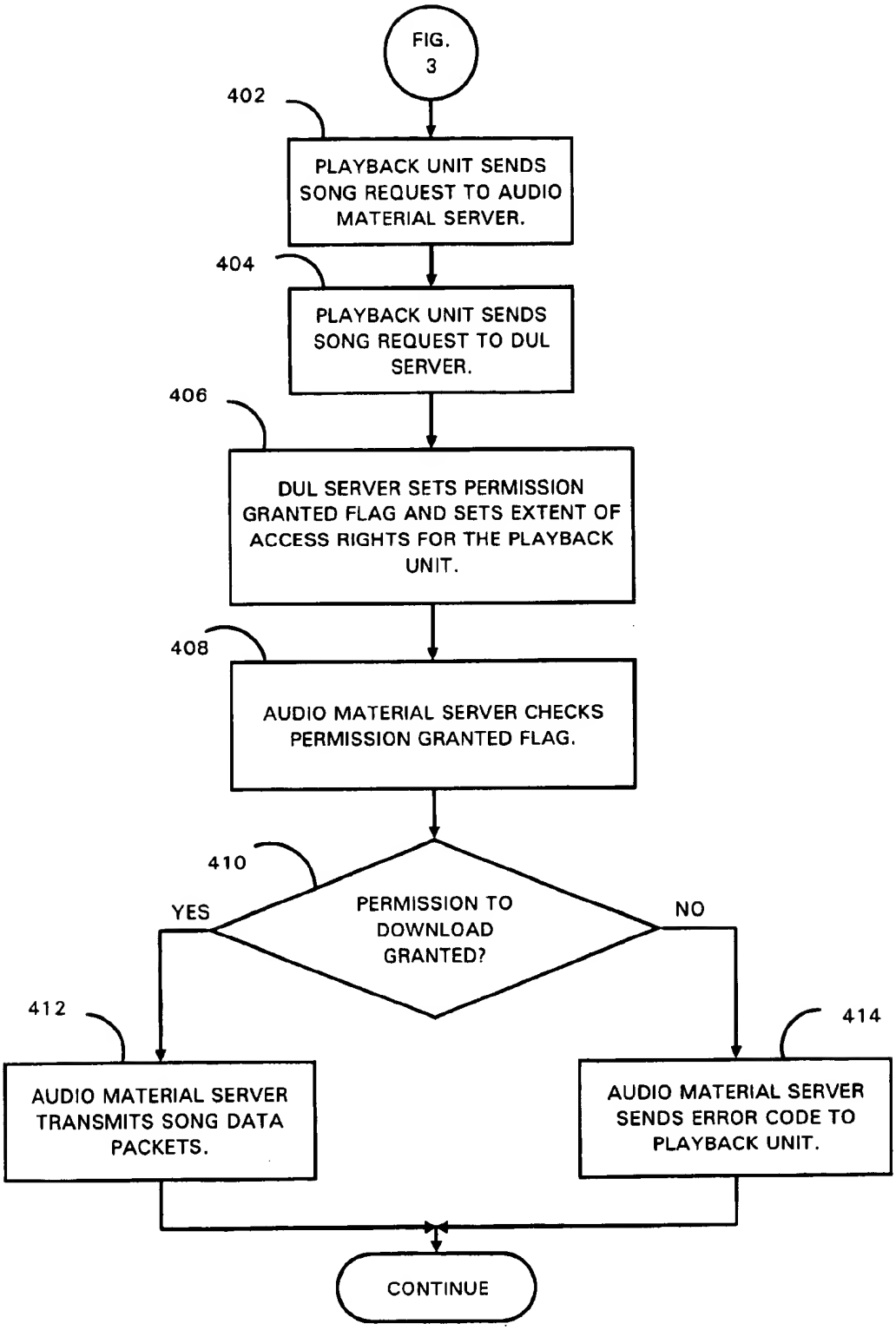


FIG. 3
402
PLAYBACK UNIT SENDS SONG REQUEST TO AUDIO MATERIAL SERVER.
404
PLAYBACK UNIT SENDS SONG REQUEST TO DUL SERVER.
406
DUL SERVER SETS PERMISSION GRANTED FLAG AND SETS EXTENT OF ACCESS RIGHTS FOR THE PLAYBACK UNIT.
408
AUDIO MATERIAL SERVER CHECKS PERMISSION GRANTED FLAG.
410
PERMISSION TO DOWNLOAD GRANTED?
YES
NO
412
AUDIO MATERIAL SERVER TRANSMITS SONG DATA PACKETS.
414
AUDIO MATERIAL SERVER SENDS ERROR CODE TO PLAYBACK UNIT.
CONTINUE

FIG. 4

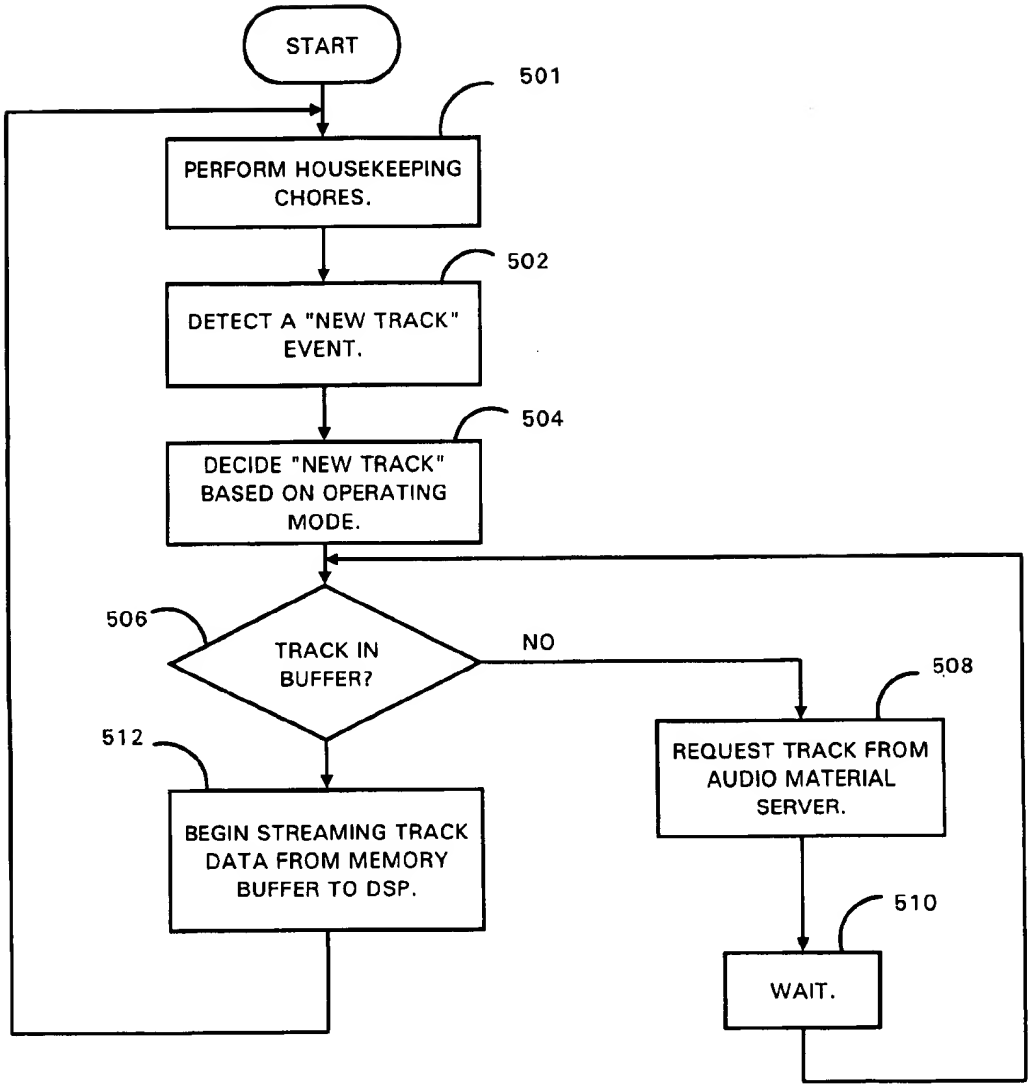

START
501
PERFORM HOUSEKEEPING CHORES.
502
DETECT A "NEW TRACK" EVENT.
504
DECIDE "NEW TRACK" BASED ON OPERATING MODE.
506
TRACK IN BUFFER?
NO
508
REQUEST TRACK FROM AUDIO MATERIAL SERVER.
510
WAIT.
512
BEGIN STREAMING TRACK DATA FROM MEMORY BUFFER TO DSP.

FIG. 5

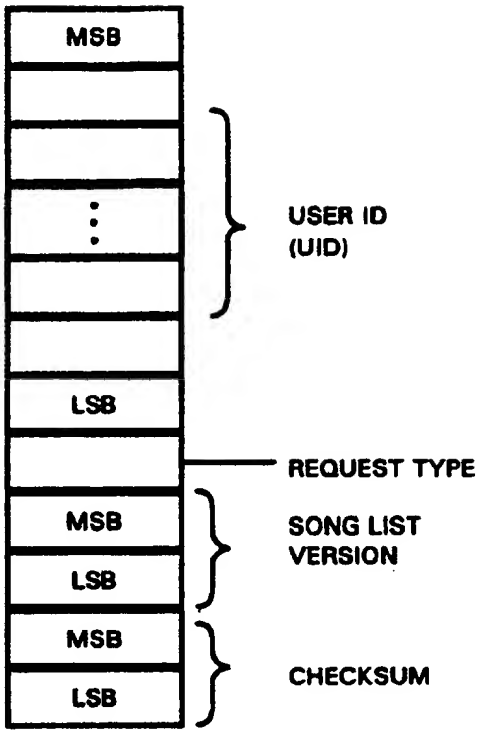

MSB
USER ID
(UID)
LSB
REQUEST TYPE
MSB
SONG LIST
VERSION
LSB
MSB
CHECKSUM
LSB

FIG. 6

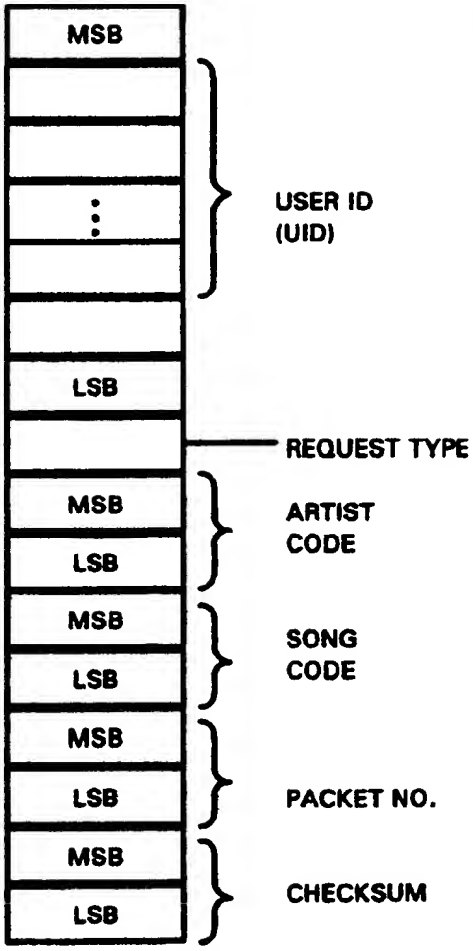

MSB
USER ID
(UID)
LSB
REQUEST TYPE
MSB
ARTIST
CODE
LSB
MSB
SONG
CODE
LSB
MSB
LSB
PACKET NO.
MSB
CHECKSUM
LSB

FIG. 7

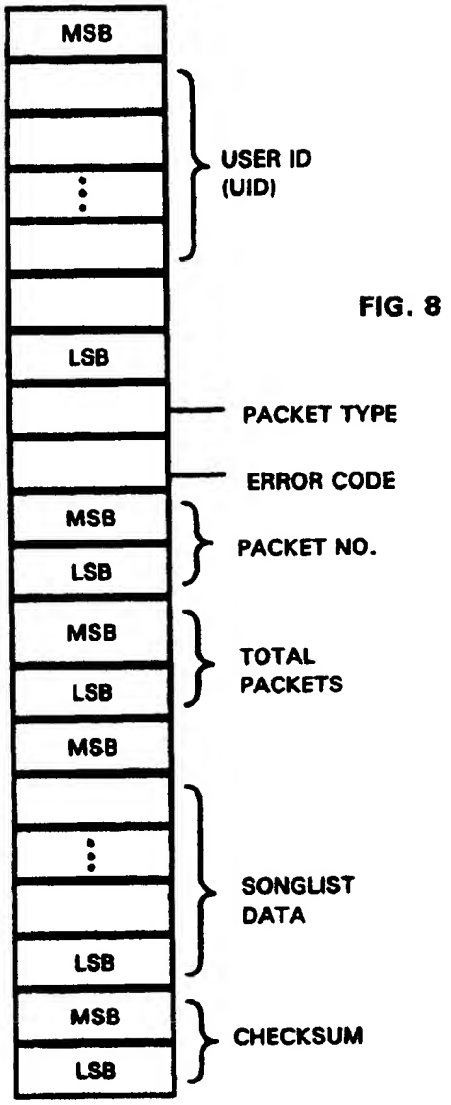

MSB
USER ID
(UID)
LSB
PACKET TYPE
ERROR CODE
MSB
PACKET NO.
LSB
MSB
TOTAL
PACKETS
LSB
MSB
SONGLIST
DATA
LSB
MSB
CHECKSUM
LSB

FIG. 8

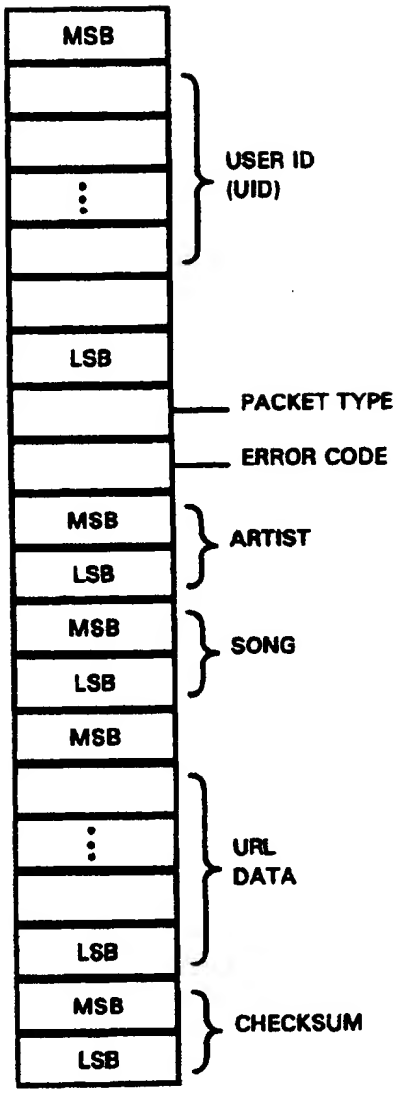

MSB
USER ID
(UID)
LSB
PACKET TYPE
ERROR CODE
MSB
ARTIST
LSB
MSB
SONG
LSB
MSB
URL
DATA
LSB
MSB
CHECKSUM
LSB

FIG. 9

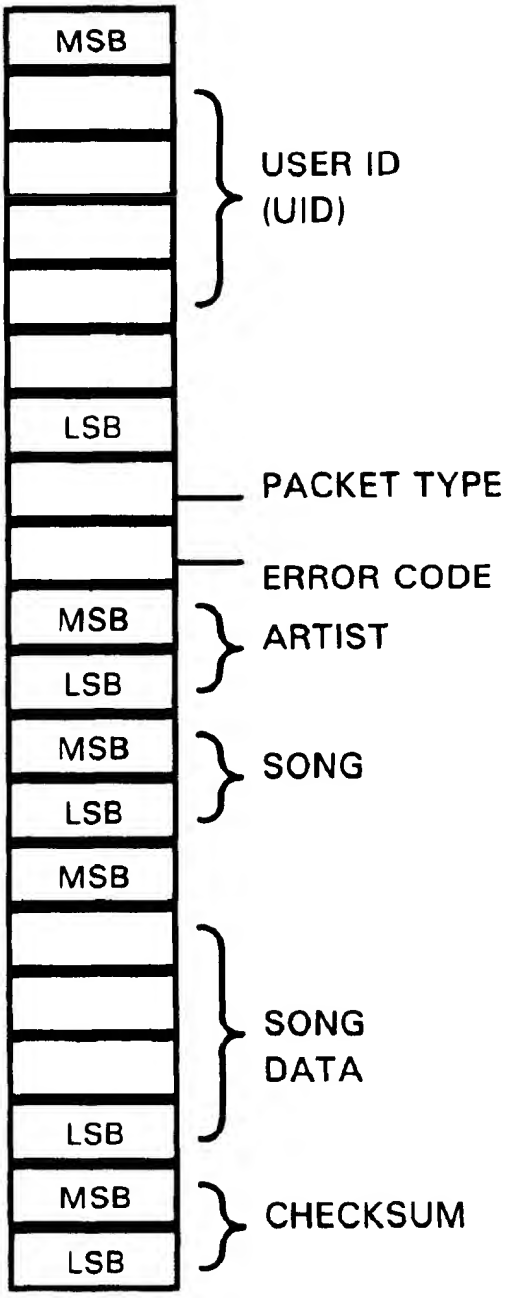

MSB
USER ID
(UID)
LSB
PACKET TYPE
ERROR CODE
MSB
ARTIST
LSB
MSB
SONG
LSB
MSB
SONG
DATA
LSB
MSB
CHECKSUM
LSB

FIG. 10

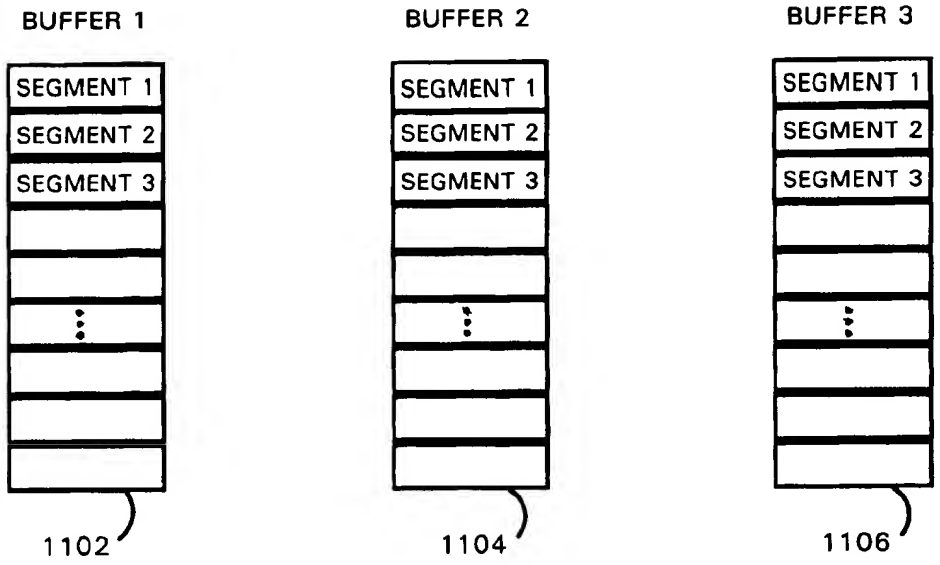

BUFFER 1
SEGMENT 1
SEGMENT 2
SEGMENT 3
1102
BUFFER 2
SEGMENT 1
SEGMENT 2
SEGMENT 3
1104
BUFFER 3
SEGMENT 1
SEGMENT 2
SEGMENT 3
1106

FIG. 11

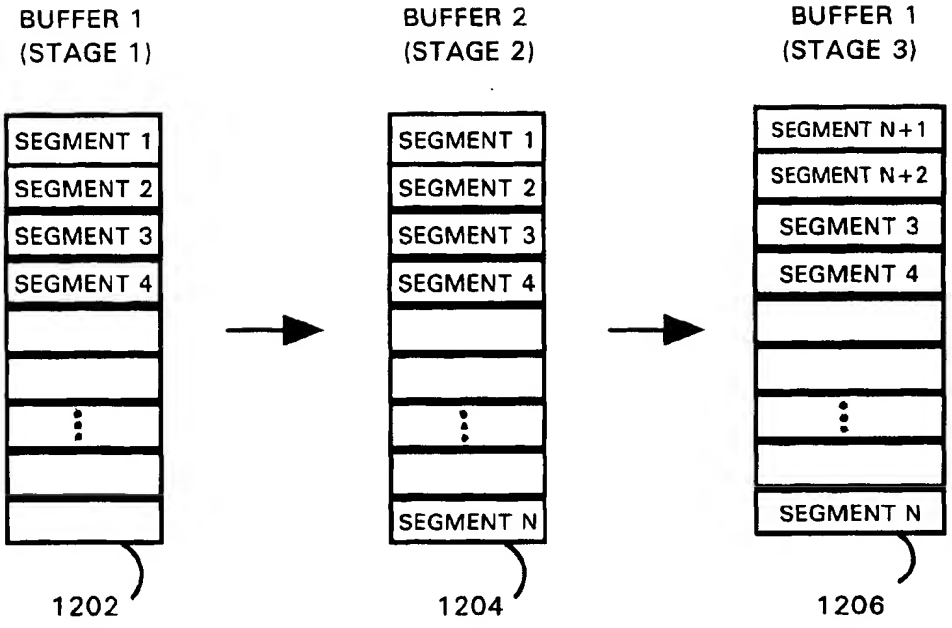

BUFFER 1
(STAGE 1)
SEGMENT 1
SEGMENT 2
SEGMENT 3
SEGMENT 4
1202
BUFFER 2
(STAGE 2)
SEGMENT 1
SEGMENT 2
SEGMENT 3
SEGMENT 4
SEGMENT N
1204
BUFFER 1
(STAGE 3)
SEGMENT N+1
SEGMENT N+2
SEGMENT 3
SEGMENT 4
SEGMENT N
1206

FIG. 12

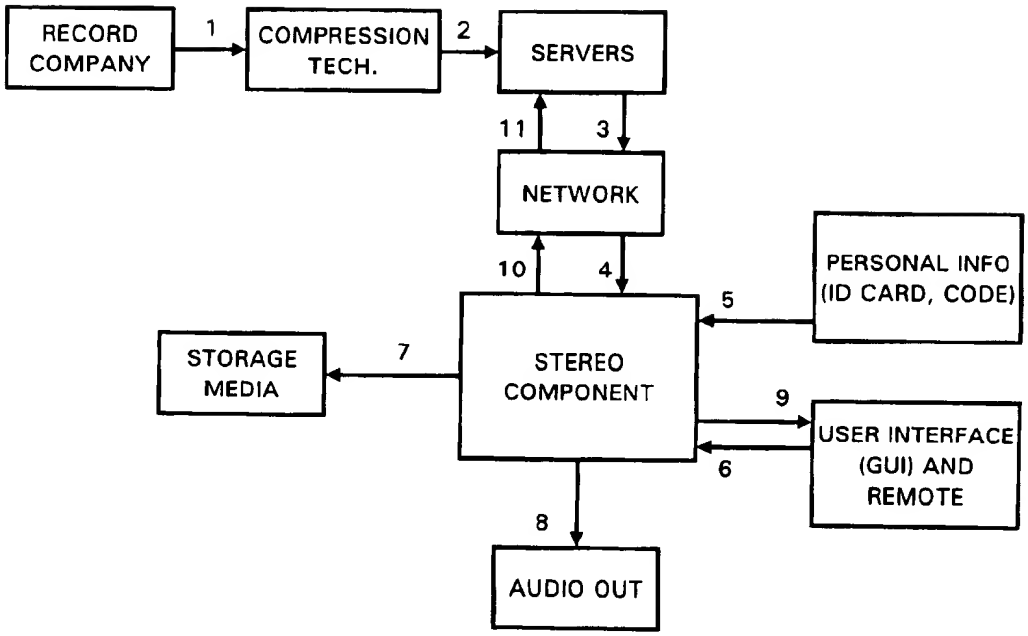

RECORD COMPANY
1
COMPRESSION TECH.
2
SERVERS
11
3
NETWORK
10
4
PERSONAL INFO (ID CARD, CODE)
5
STORAGE MEDIA
7
STEREO COMPONENT
9
USER INTERFACE (GUI) AND REMOTE
6
8
AUDIO OUT

FIG. 13

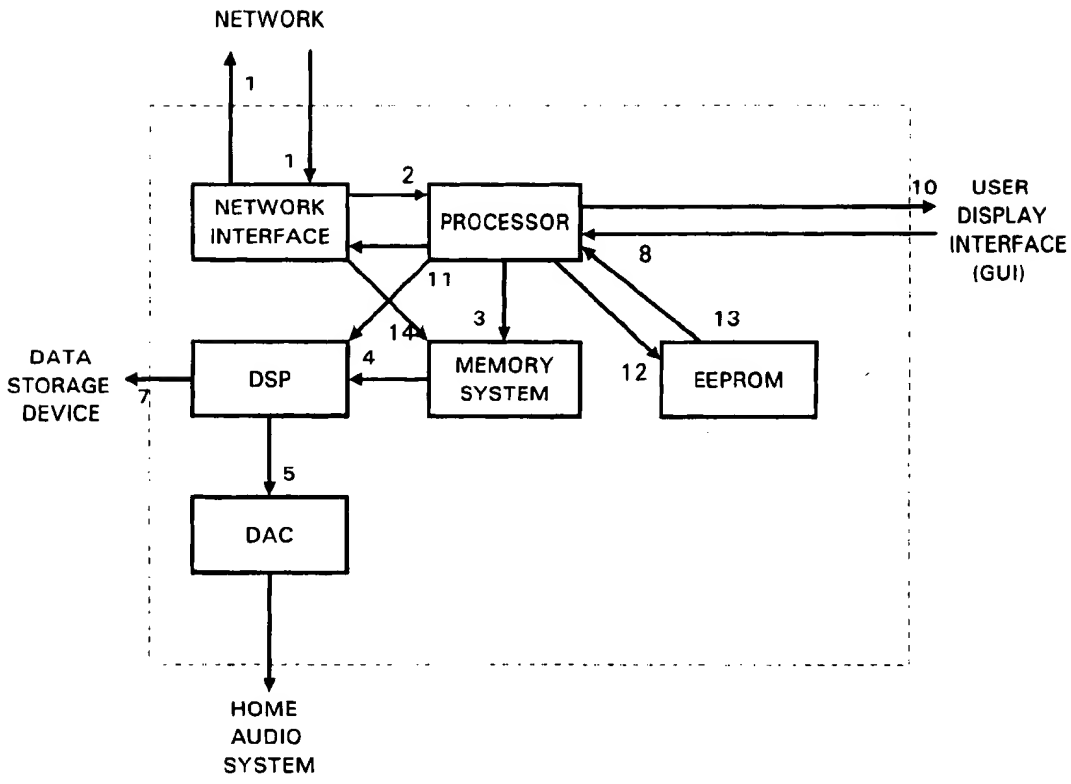

NETWORK
1
1
NETWORK INTERFACE
2
PROCESSOR
10
USER DISPLAY INTERFACE (GUI)
8
11
3
13
14
DATA STORAGE DEVICE
7
DSP
4
MEMORY SYSTEM
12
EEPROM
5
DAC
HOME AUDIO SYSTEM

FIG. 14

SYSTEM FOR PLAYBACK OF NETWORK AUDIO MATERIAL ON DEMAND

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to music playback systems and, more particularly, to playback of network audio material in response to user command.

2. Description of the Related Art

Two popular means of listening to digitally encoded audio material are conventional home audio music playback systems that include conventional media players that reproduce recorded music information and computer-based systems that typically include a standard personal computer (PC) or similar machine capable of utilizing a variety of digital music formats, including pre-recorded media and computer audio files. Both types of systems permit users to initiate playback of a selected piece of audio material, such as recorded songs or other music.

Conventional home audio music systems typically include a player that accepts media encoded with digital audio material. Such media include the compact disc (CD), MiniDisc (MD), and digital audio tape (DAT) formats. The CD format comprises a plastic-coated aluminum substrate from which digital audio material can be optically retrieved. The MiniDisc is a magneto-optical storage format. The DAT format comprises a tape substrate with a magnetic recording layer in which digital audio material is magnetically recorded. The CD format is the most popular current means of delivering recorded music and offers the largest library of recorded works for selection. Other popular media for playback of digital music information include the "Laserdisc" (LD) format and the "Digital Video Disc" (DVD) format, both of which can combine video information with music or other digital audio information. All of these formats offer a relatively stable recording media, high quality audio reproduction, convenient storage and playback, and simple operation of players.

Home audio players, such as CD players and DAT players, can provide exceptional quality sound reproduction, made all the better because such players are typically connected to a relatively good quality, home high-fidelity music system. The CD format discs are convenient because they are especially easy to store and take up comparatively little storage space. Playback of CDs also is convenient, because the CD player is ready to read the digital audio material upon power-up of (application of electrical power to) the player. For playback the discs are simply inserted into a CD player's tray or slot and started with simple one-button operation. In addition, such home music systems are typically arranged in a comfortable setting within the home. Such home music systems typically include, in addition to the CD player that reads the digital audio material and produces a playback signal, one or more amplification and control devices, signal processors, and power amplifiers to process and amplify the analog playback signal, and also a set of loudspeakers, to receive the amplified playback signal and convert it to sound.

Home music systems permit a user to initiate playback on demand by the selection of an appropriate disc or tape media. The selection, however, must be made from the user's personal collection of media on hand, which limits the available music to that which the user has purchased, borrowed, or otherwise received. This limits the repertory from which the user may select and discourages many users

from review of and experimentation with audio material and musical products. This is undesirable from the perspective of the music industry, because it is believed that such experimentation and review can lead to further sales of recorded audio material. Borrowing media from another user or from a commercial enterprise, thereby expanding the library of material available to include that which is maintained by acquaintances or rental shops but this is not convenient.

In contrast to the home audio system with CD or DAT player, the conventional computer-based system with appropriate software and hardware can provide music either from pre-recorded digital media or from computer audio files. For purposes of this discussion, the computer-based playback system will be referred to as a PC-based system, regardless of the computer on which it is based.

If the PC-based system includes a CD-ROM drive and sound card, for example, a CD with digital audio material can be inserted into the drive and the sound recorded on the CD can be listened to through PC speakers that receive output from the sound card. This mode of listening has the same limitations of repertoire as the home audio CD player. Moreover, the typical PC-based system does not have audio components as good as that of the typical home audio system, and is usually not located in as comfortable a setting as the typical home audio system.

A PC-based system with access to a network such as the Internet can, with the appropriate software, download audio material for playback. This audio material can comprise, for example, digitized sound clips stored as ".wav" files, MPEG (Motion Picture Experts Group) Audio Layer 3 (MP3) compressed-audio files, streaming audio formats for continuous play of audio material, and other digital formats for the storage of audio material, all of which can be stored on a fixed media and received by the PC. More recently, another sound file format called the Secure Digital Music Initiative (SDMI) has been proposed. Alternatively, the audio material can be received from a network file server, and then stored on the hard drive of the PC itself. Additional software can be used for convenient organization of downloaded music files. Other audio material may comprise streaming audio files, which require additional streaming audio playback software.

Such network downloading of music can vastly expand the repertory from which the user may select, and encourages review of and experimentation with audio material. Again, however, the PC-based system provides limited enjoyment because the typical PC-based system does not have audio components as good as that of the typical home audio system, and is usually not located in as comfortable a setting as the typical home audio system. Furthermore, the PC-based system is not as convenient to use as the home audio system, because the PC is typically located in a work environment away from the home audio system, and the operating system of the PC requires an initial lengthy boot-up process that loads an operating system from peripheral storage, the launching of appropriate player software, and the navigation of a potentially complicated software interface with multiple windows and drop-down menus to select before initiating playback each time the user wants to listen to audio material.

In addition, operating a PC-based system, gaining Internet access, and downloading audio files can require computer skills not possessed by the average listener, in addition to requiring the initial purchase of the computer equipment. Peripheral playback devices also may need to be installed on the PC-based system, requiring knowledge of the operating system and peripheral interface, and some of these formats

only provide low-fidelity playback that is adequate for audio while working at the computer, but is not useful as an adjunct or replacement for the home audio system and CD player.

Some forms of PC-based systems also are meeting with resistance from commercial music industry interests and from artists because of the potential for widespread copyright violation and the difficulty of policing the download and duplication of audio information files by users. The availability of network databases and the download and duplication of audio files make it almost impossible to monitor and control the distribution of recorded musical performances. Some PC-based systems also may be problematic in view of governmental regulation, such as the Audio Home Recording Act passed by the U.S.A. legislature, which under certain conditions mandates a serial copy management system (SCMS) to control digital copying. It would be advantageous to provide a system that is capable of interfacing with home audio systems for high quality playback, that has access to the large repertory possible through network databases, and would have the acceptance of commercial music interests and artists.

From the discussion above, it should be apparent that there is a need for a system that can provide playback of a wide range of audio material on demand, using the home audio system for high quality playback, without requiring sophisticated computer skills, and with controlled access to audio material and controlled distribution and duplication of the material. The present invention fulfills this need.

SUMMARY OF THE INVENTION

The present invention provides a system for playback of network audio material on demand by using a playback apparatus that provides an interface to network audio files that are retrieved in real time in response to user selection. In accordance with the invention, the playback unit provides an interface between a conventional home audio system and a network source for audio material, such as the Internet. The playback unit has a relatively simple built-in operating system that is not accessed from peripheral storage, does not require a lengthy boot-up sequence, and cannot be manipulated without the authorization of the manufacturer or network source. As a result, the playback unit can be operated without special computer skills or navigation of complicated PC-like windows. Receipt of audio material and enforcement of distribution rights can be controlled by network servers that provide the audio material to the playback unit. In this way, the playback unit can retrieve a wide range of digital audio material from the network on demand, thereby vastly expanding the range of music available for playback, can reproduce that music using the home audio system for high quality playback in a comfortable setting, and can provide controlled access to audio material and controlled distribution and duplication of the material.

The playback unit includes a user interface and display component, which presents an easy-to-use interface that simulates playback controls that might be found on a conventional player such as a CD player or DAT player. The user interface and display component substantially duplicates the appearance of a conventional home audio player control panel, such as CD player buttons and track displays. The playback unit also includes memory for holding program instructions and temporarily storing audio material for playback so it is not accessible to the user, and includes a microprocessor that controls operation of the playback unit. In one aspect of the invention, the playback unit includes a network interface to communicate with the network, send user commands, and receive audio material. The network interface can communicate using a number of different protocols having a variety of physical connection schemes, such as telephone line modem connections, high-speed Ethernet connections, and cable modem connections. The playback unit also includes an output interface that receives the audio material and provides it to the home audio system in a format that can be reproduced by that system.

Other features and advantages of the present invention should be apparent from the following description of the preferred embodiment, which illustrates, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a playback unit constructed in accordance with the present invention showing the connections to a home audio system and a network.

FIG. **2** is a representation of the screen display shown on the user interface of the playback unit illustrated in FIG. **1**.

FIG. **3** and FIG. **4** are processing flow diagrams that illustrate the processing steps executed by the components illustrated in FIG. **1** to request, receive, and play audio material from the network.

FIG. **5** is a processing flow diagram that illustrates the processing steps executed by the playback unit processor illustrated in FIG. **1**.

FIGS. **6**, **7**, **8**, **9**, and **10** are representations of packet information processed by the playback unit illustrated in FIG. **1**.

FIG. **11** is a representation of the buffers contained in the memory illustrated in FIG. 1.

FIG. **12** is a representation of the loop buffering operations executed under control of the microprocessor illustrated in FIG. **1**.

FIG. **13** is a data flow diagram of the FIG. **1** system operation, showing the information that is transmitted among the system components.

FIG. **14** is a data flow diagram of the playback unit operation, showing the information that is transmitted among the playback unit components.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 illustrates a playback unit **100** constructed in accordance with the present invention. The playback unit communicates over a network, such as the Internet **102**, to request digital audio material from one or more audio material servers **104**. The playback unit receives audio material from an audio material server and provides it to a conventional home audio system **106** for playback. The playback unit **100** has a simple operating system that accesses instructions from high-speed semiconductor memory, does not require a lengthy boot-up sequence, and cannot be manipulated by the user. Thus, the playback unit does not require the user to launch special software such as the "Windows 98" operating system by Microsoft Corporation to initiate playback, and therefore the playback unit is very stable in operation and can be operated without special computer skills or navigation of complicated PC-like windows. Access to the audio material and authority for distribution rights are preferably controlled by a directory and user list (DUL) server **107** described further below. In this way, the playback unit **100** can retrieve a wide range of digital audio material from the network upon user demand,

thereby vastly expanding the range of music available for playback, and can reproduce that music using the home audio system for high quality playback in a comfortable setting.

The playback unit **100** is most likely to be installed adjacent to the home audio equipment **106**, which typically includes a variety of amplifier, processor, receiver, control, and record/playback units. The playback unit **100** comprises a stand-alone device that is preferably the same size as the individual home audio system devices, so as to be physically and aesthetically compatible with them. The playback unit includes a network interface **110** that provides a communication channel with the Internet **102** and to the audio material server **104**. The network interface can communicate using a number of different protocols having a variety of physical connection schemes, such as telephone line modem connections, high-speed ISDN and Ethernet connections, and cable modem connections.

Playback Unit Components

The playback unit **100** includes a user interface and display component **112**, which presents an easy-to-use interface that substantially duplicates the appearance of typical user-operable controls that might be found on a conventional home audio player that plays physical media, such as a CD player or a DAT player. These controls may include, for example, PLAY, STOP, FORWARD, BACKWARD, PAUSE, TRACK, and SELECT buttons. In the preferred embodiment, the user interface and display component **112** includes a touch panel or screen that responds to user activation of virtual buttons shown on the display screen. The function represented by the activated display button is then executed by the playback unit. The touch panel permits easy updates to the player functionality by changing the buttons and their operation with new program instructions stored in memory, as described below. Alternatively, the buttons may comprise actual physical buttons that have an electromechanical interface so they respond to physical pressure by producing a signal that activates the corresponding function.

FIG. **2** shows an exemplary display interface comprising a touch panel screen **202** of the playback unit **100**. The playback unit preferably includes at least one physical button, a power button **204** that initiates the application of electrical power to the circuits of the playback unit. The playback unit may also include a sensor, such as an infrared sensor **206**, for receiving command signals from a remote control unit (not illustrated). The display interface has a display area **208** on which playback status information is shown. For example, FIG. **2** shows the display area **208** with a list of song or selection name, track number, artist name or disc (compilation), and song playing time. The display interface may include virtual operation buttons, or actual physical buttons, that cause operations such as reverse **210**, pause **212**, play **214**, stop **216**, forward **218**, fast forward/skip **220**, cursor navigation up **222** and down **224**, and a function select **226** button. As noted above, the buttons **210–226** may be virtual buttons of a touch panel screen **202** also having a status information display area **208**, or may be physical buttons adjacent a display area **208** in which alphanumeric information is shown.

Returning to the illustration of FIG. **1**, the playback unit **100** also includes an output interface **114**, memory **116**, and microprocessor **118**. The output interface processes the audio material and provides it to the home audio system in a format that can be used by that system. The connection to the home audio system **106**, for example, can comprise a direct wire connection to home audio loudspeakers that receive an analog signal, or can be a connection to a signal processor, receiver, or other control and/or amplification device for playback using the loudspeakers of the home audio system. The memory **116** holds data including program instructions and temporarily stores audio material for processing and playback. The memory may comprise a combination that includes, for example, semiconductor memory such as electrically erasable programmable read only memory (EEPROM) or flash memory for holding program instructions and buffer memory for holding song data (audio material).

The program instructions are automatically executed by the microprocessor **118** when power is applied to the playback unit. Thus, there is no need to access an operating system stored on a disk drive or other peripheral storage device to operate the playback unit. As a result, the playback unit does not require an electromechanical storage device (such as a disk drive), is very stable in operation, and does not require a boot-up sequence. The buffer memory for audio material storage is preferably dynamic random access memory (RAM), which is a low-cost, efficient means of temporarily storing digital audio material to be processed for playback. In addition, the volatility of the buffer memory ensures that the user has no permanent copy of the audio material, thereby ensuring protection of copyrighted material. As described further below, storage of the audio material in the memory is determined by data downloaded through the network interface **110**, and therefore is externally controlled.

The playback unit **100** operates under control of the microprocessor **118**, which controls operation of the other playback unit components **110**, **112**, **114**, **116**. The microprocessor also performs the various calculations and computations required for processing the audio material and preparing it for playback. If desired, the microprocessor component **118** may work along with a specialized digital signal processing (DSP) circuit for performing sound data computations and, if necessary, audio material data decompression. As noted above, the program steps executed by the microprocessor are stored in a program instruction flash memory portion of the memory **116**. Therefore, although the user cannot change the operating system instructions, the playback unit operation is fully determined by the stored program instructions, which can be changed by loading new instructions into the memory **116**. This permits changing, for example, the display buttons to provide new functions.

Playback Unit Operating Steps

FIG. **3** is a flow diagram of the processing steps executed by the microprocessor **118** of FIG. **1**, and illustrates the processing carried out by the playback unit **100** in response to user commands. An initial step, as represented by the flow diagram box numbered **301**, occurs when electrical power is applied to the playback unit. As noted above, the operation of the playback unit is sufficiently simple so that no operating system loaded from peripheral storage is required, therefore, there is no boot sequence, and the user cannot alter system operation of the playback unit. As a result, upon the application of electrical power, the playback unit **100** is immediately operational.

In the first operational step, represented by the flow diagram box numbered **302**, the user selects a music category or type of song desired for playback from a list. This list may include categories such as the artist, the song title,

the album, and musical genres. In addition, the user may limit search results by confining the query to specific, user-defined categories. The generated list appears on the display area of the user interface. In the next step, the playback unit sends the version of the current song list to the directory and user list (DUL) server **107**, shown in FIG. 1. During this step, the DUL server also can perform user list checks and authorization confirmation, if desired. In this way, the DUL server acts as a "gatekeeper" to ensure that only appropriate users are being granted access to the audio material, thereby ensuring commercial music interests and artists have desired control over distribution. The flow diagram box numbered **304** represents this operational step.

At the decision box numbered **306**, the DUTL server checks to determine if the received song list is current. If the song list is not current, a negative outcome at the decision box **306**, then a new song list is available and the server sends back an updated song list, as represented by the flow diagram box numbered **308**. If the playback unit song list is already current, an affirmative outcome at the decision box **306**, then no song list data transmission from the DUL server is needed. With a confirmed current song list, the user is now permitted to select a track from among those available in a selection menu. The selection menus are displayed, for example, on the display area of the interface illustrated in FIG. **2**. The user may need to scroll up and down the displayed selection menu list. Tracks can be selected by artist, genre, disc name, or a number of other factors. The operation of a user making an artist and song selection is represented by the flow diagram box numbered **310**. At the next step, represented by the flow diagram box numbered **312**, the playback unit sends the user-requested song title information to the DUL server. The DUL server returns the network address for the requested song. This step is represented by the flow diagram box numbered **314**. The playback unit is now ready to retrieve audio material from the network. The flow diagram for these operations continues in FIG. **4**.

In the case of an Internet network connection, the returned network address is referred to as the uniform resource locator (URL) for the song. Once the song URL is received, the playback unit initiates communication with the appropriate audio material server to request the song from the appropriate directory. This step is represented by the FIG. **4** flow diagram box numbered **402**. In the preferred embodiment, the DUL server maintains control over communication from the playback unit to the network, and therefore the DUL server can determine if the audio material server at the indicated URL is inactive or not responding. If either is the case, then the DUL server will detect this condition and may send the URL of a backup or alternate audio material server at which the requested song is stored. In this way, the user may still gain access to the requested song and listen to it.

When the playback unit sends the song request to the server whose URL it received from the DUL server, it also sends a user identification code (user ID) and encrypted password information to the DUL server. This step is represented by the flow diagram box numbered **404**. That is, because the DUL server maintains communication control, the DUL server can perform a gatekeeping function to permit or prevent the playback unit from receiving the requested audio material. If the user ID and password information is validated, then the DUL server sets a permission granted flag that is checked by the appropriate audio material server. The permission granted flag may be stored at the DUL server and remotely checked by the audio material server, or otherwise communicated to the audio material server. This operation step is represented by the flow diagram box numbered **406**.

The permission granted flag dictates whether or not a user will be permitted to download a song for listening and also for recording. Other authorizations, in accordance with the Secure Digital Music Initiative (SDMI) for example, may be accommodated. That is, the permission granted flag may grant or deny a range of distribution, reproduction, copy, and recording rights. Thus, the permission granted flag may include a copy authorization flag to control digital copying. These rights may be granted in accordance with predetermined arrangements between commercial music interests and artists on the one hand, and entities controlling the DUL server and audio material servers on the other hand. If permission to record is granted, for example, a feature of the output interface will be set to permit an output format that is suitable for recording. If the user is not granted recording privileges, then no output will be available at the needed output connections of the output interface, or any attempt to exceed the granted song rights will result in display of an error message on the display interface and a halt to operations of the playback unit.

Thus, after the permission granted flag is set by the DUL server at step **406**, the audio material server checks the flag at step **408**. The flag may be sent to the audio material server by the DUL server or forwarded by the playback unit. If the permission granted flag indicates that the user has been granted permission to download the requested song, an affirmative outcome at the decision box **410**, then at the flow diagram box numbered **412** the audio material server transmits the audio material (comprising a sound file or streaming audio information) to the playback unit, where it is received by the network interface as described above. Other operation of the playback unit then continues. If the permission granted flag indicates that the user has not been granted permission to download the song, a negative outcome at the decision box **410**, then at the flow diagram box numbered **414** the audio material server sends an error code to the playback unit to halt operation. Similar processing will be performed if other user actions are attempted that require authorization, such as digital copying.

Playback Unit Processor Operating Steps

FIG. **5** is a processing flow diagram of the processing steps executed by the playback unit microprocessor **118** shown in FIG. **1**, and illustrates the processing repeatedly executed by the microprocessor during operation for responding to user commands. During operation, the microprocessor executes "housekeeping" chores as part of typical background processing as indicated by the flow diagram box numbered **501**. The housekeeping chores include, for example, updating the user display, scanning the user keypad buttons for actuation, scanning any infrared receiver for user input, and downloading tasks. The downloading tasks may include, for example, downloading the first few seconds of each track on a current selected disc to reduce the latency time when one of the tracks is later selected by the user. Such samples from the tracks may be stored in the memory of the playback unit for later listening. FIG. **5** illustrates processing when a "New Track" event is detected.

The detection of a user action at the user interface and display is represented by the first FIG. **5** flow diagram box numbered **502**, which indicates that the processor detects a "New Track" event. A "New Track" event is defined to be a user action such as selection of the Play button, Skip Track

button, or other button such as "Jump Track" or "Change Disk" or the like. Upon receipt of a "New Track" event, the microprocessor determines the system operating mode, which may comprise either a normal mode, random mode, or custom mode. This step is represented by the flow diagram box numbered **504**.

The system operating modes specify an algorithm for determining the next track. The normal mode specifies that the "next" track is the next sequential track in the selected compilation. The random mode specifies that the "next" track is a randomly selected track in the selected compilation. The custom mode specifies that the "next" track is a user programmed track, such as when a user records a program of track selections for playback in the programmed order.

Once the "next" track is determined in accordance with the operating mode, the microprocessor determines if the next track is in the memory buffer of the playback unit. If the next track is not in the buffer, a negative outcome at the decision box **506**, then the microprocessor requests the missing track from the appropriate audio material server. The request is represented by the flow diagram box numbered **508**. The microprocessor waits for receipt of the missing track, as indicated by the flow diagram box numbered **510**, and then loops to the decision box **506** again. Once a sufficient portion of the track is received or otherwise located in the buffer, an affirmative outcome at the decision box **506**, the microprocessor begins streaming the audio material data from the memory buffer to the DSP for processing. The DSP may be one of a number of commercially available DSP engines for the decompression and decoding processing of digital audio data. Those skilled in the art will understand the processing involved without further explanation.

Data Packets

As noted above, the network interface can communicate using a number of different protocols having a variety of physical connection schemes. FIGS. **6**, **7**, **8**, **9**, and **10** show the data bytes of exemplary data packets that can be used to communicate the different types of information needed for operation of the system constructed in accordance with the present invention. FIG. **6** illustrates the data packet when the playback unit requests a song list version check. FIG. **7** illustrates the data packet used when the playback unit requests a song from the audio material server. FIG. **8** illustrates the data packet used when the DUL server sends an updated song list to the playback unit. FIG. **9** illustrates the data packet when the DUL server sends the URL of a song to the playback unit. FIG. **10** illustrates the data packet when an audio material server sends a song to the playback unit. It should be understood that the data packets of FIGS. **6** through **10** are intended only to illustrate the type of information that may be exchanged between the playback unit, DUL server, and audio material server, and should not be taken in a limiting sense as a requirement for operation of a system constructed in accordance with the present invention.

FIG. **6** shows the user request data packet. With this packet, the playback unit requests a check to ensure the song list being used is current. The first eight bytes comprise an optional user ID field that would be sent so the DUL server can identify who is requesting the song list. The next byte is a request field that permits the DUL server (or any other network server) to identify what data is being requested by the client playback unit. The next two bytes comprise a song list version field that provides the version number of the song list currently stored in the memory of the playback unit. Finally, the last two bytes of the data packet contain checksum data for identifying any errors that might occur during transmission over the network connection.

FIG. **7** shows the song request data packet. The first eight bytes contain the optional user ID data and the next byte contains the request data, as described above for FIG. **6**. The next two bytes contain an artist code comprising a unique identification number for an artist of a song. The artist code (AC) can be used as an index to the directory on the audio material server that contains the artist's work. The next two bytes contain the song code (SC) comprising a unique identification code for the requested song. The song code can be used as an index to the song data file on the audio material server. The next two bytes contain a packet identification code (packet ID) that tells the audio material server which packet to send next. That is, because each song to be downloaded is sent in several pieces, the playback unit must be able to communicate to the audio material server which portion of the song is needed next. The last two bytes comprise the checksum field for identifying errors.

FIG. **8** shows the updated song list packet, which is sent from the DUL server to the playback unit. The first eight bytes contain the user ID, which in this case is returned to the playback unit for purposes of error checking, to confirm that the updated song list information was sent to and received by the correct playback unit. The next byte contains packet type data, which is necessary to let the playback unit know how to interpret the data it receives. That is, the packet type data for this transmission lets the playback unit know that an updated song list is being sent. The most significant bit (MSB) position indicates whether or not this is the last packet of data that will be sent with this type of information. An error code (EC) byte is next, which provides the error code that (if necessary) is displayed on the display component. A two-byte packet ID field is next, and is used to let the playback unit know which area of the memory in which the data needs to be stored. That is, the song list data is organized according to a predetermined arrangement and the playback unit needs to conform its memory to that predetermined arrangement. Next is a two-byte field that provide the total number of packets to transmit the song list update information. This is needed so the playback unit can adjust the buffering scheme. The next field has a variable number of bytes, as it contains the songlist data. This number may vary depending on the network connection and the transmission protocol being used, among other considerations that will occur to those skilled in the art. Finally, the last two bytes of the updated song list packet comprise checksum data for identifying errors.

FIG. **9** shows the packet containing the URL of a requested song. The first eight bytes contain the user ID, which as before is returned to the playback unit for purposes of error checking, to confirm that the requested information was sent to and received by the correct playback unit. The next byte contains packet type data, which is necessary to let the playback unit know how to interpret the data it receives. In this case, the packet type data will indicate that a URL is being sent, and the MSb position will indicate whether or not this is the last packet of data that will be sent with this URL information. An EC byte is next, to provide any error code to be displayed on the display component. The next two bytes are an artist code (AC) that is sent to the playback unit to ensure that it can place the URL data in the correct memory buffer. A song code (SC) occupies the next two bytes, the code being used to ensure that the playback unit

places the URL data in the correct memory buffer. The next four bytes contain URL data that identifies the audio material server that contains the desired song for playback. The last two bytes of the URL packet contain the checksum data for identifying errors that occur during transmission.

FIG. **10** shows the song data sent by an audio material server to the playback unit. The user ID is contained in the first eight bytes, returned as a form of error checking and to confirm that the requested song information was sent to and received by the correct playback unit. The packet type data is contained in the next byte, and in this case the MSb indicates whether or not this is the last packet that will be sent. The EC is contained in the next byte. The AC occupies the next two bytes, to ensure that the playback unit places the song information in the correct memory buffer. The SC occupies the following two bytes, again sent to ensure that the playback unit places the song information in the correct buffer. The next field contains the actual song data and has a variable number of bytes. The number of bytes of song data may vary depending on the network connection and the transmission protocol being used, among other considerations that will occur to those skilled in the art. The song data will be placed in a section of memory that depends on the artist, song, and packet number. Finally, the last two bytes of the song data packet comprise checksum data for identifying errors.

Memory Buffering Control

FIG. **11** is a representation of the buffering operations of the playback unit. The playback unit memory may be segregated into a number of sequential buffers, with each buffer preferably containing one song. Based on typical compression algorithms, the size of each buffer will be approximately two megabytes (2 MB). The number of buffers that can be accommodated by the playback unit is determined by the amount of memory (bytes) that the playback unit microprocessor can access, so the number of buffers available will be variable. Nevertheless, the functionality of the playback unit remains the same regardless of the available memory address space. The addressable portions that make up each buffer will contain data that, when processed, produces approximately ten seconds of audio listening.

The buffering of song data ensures that a song may be downloaded and temporarily stored in less time than needed to play the song. The speed of this buffering operation will depend on the speed of the network connection available. Buffering begins after the user selects one or more songs for listening. The playback unit downloads the selected songs in data packets, which in the preferred embodiment each contain approximately ten seconds of compressed digital audio information. As noted above, the number of data packets to be downloaded for each song is an undetermined number that depends on the song length. It is not necessary for song data download to be completed for one song before download for another song can begin. Preferably, portions of each selected song will be downloaded as the first one begins to play. This is illustrated in FIG. **11**, which illustrates the multiple buffers **1102**, **1104**, **1106** into which the memory of the playback unit is segregated. The current song's buffer has priority over all other buffers; the data flow into this buffer is maintained such that continuous playback of the song is guaranteed. The buffers corresponding to the following musical selections are periodically updated with their songs' data.

For example, if a user wants to hear Song1, Song2, and Song3, the playback unit downloads a number of packets for Song1 into the first available buffer **1102**. Once a sizeable amount of compressed audio information is stored for that song, the playback unit begins to process the information and play the song, providing the processed information to the home audio system. The audio information will be processed by the microprocessor and sound data DSP, if one is included in the playback unit. The amount of stored information needed before playback begins will depend on the microprocessor, DSP, and other sound components of the playback unit. As the first song (Song 1) is being played, the playback unit continues to operate and, in background operation, continues to download the Song 1 data into the first buffer **1102**, and also downloads data for the other selected songs into the other buffers **1104**, **1106** in an alternating fashion. Each song will be placed into a different sequential buffer. This ensures that some portion of each selected song will be downloaded and available as soon as possible, thereby permitting the user to skip to one of the other selected songs after playback has begun.

The playback unit preferably performs a loop buffering operation, which is illustrated in FIG. **12**. The loop buffering operation progresses from left to right in FIG. **12**. Loop buffering is used to limit the size needed for each buffer. In particular, a buffer is not expected to have sufficient capacity to contain the entire data needed for one song. Rather, data in a given buffer is overwritten as it is processed and played. Thus, after the last segment of memory in a buffer for a song has been filled with a song data packet and that buffer is processed for listening, the next song data packet will be written to the first segment in that buffer. As described above, this writing will be directed by setting the various fields in the song data packet illustrated in FIG. **9**.

Three exemplary stages of loop buffering are illustrated in FIG. **12** from left to right, showing the contents of a buffer at a first stage **1202** of buffering, at a second stage **1204** of buffering, and at a third stage of buffering **1206**. As the song is downloaded at the first stage **1202**, the buffer is filled with data bytes for Segment 1, Segment 2, Segment 3, and so forth. Eventually, the last available segment in the buffer is filled **1204** with Segment N, before the song has been completely downloaded. Therefore, the next segments of incoming data, Segment N+1, Segment N+2, Segment N+3, and so forth, will overwrite the prior data in the buffer **1206**. Buffering will continue looping in this fashion, overwriting repeatedly until the song is completely downloaded.

Loop buffering ensures that the user can scan a song in a backward direction, such as might be done for review to hear missed lyrics or other reason. If a user decides not to listen to the current song and skips it entirely on playback, it remains in the playback unit memory so the user can return to the skipped song and listen to it. In all cases, loop buffering and overwriting of buffer data will not begin until the first segment of the buffer has been processed for listening. That is, listening to a song cannot begin until the buffer for that song is initially filled, and overwriting will not begin until listening to that song has begun. However, if the user adds more songs to the playback unit than can be downloaded into the available memory, then a newer song on the playback list will begin overwriting the oldest song in memory after the last segment of the last available buffer is filled with song data.

System Data Flow Diagram

FIG. **13** is a data flow diagram of the system of FIG. **1**, showing the information that is transmitted among the system components and how those components interact. In

a preliminary stage of information flow, music is submitted for digital conversion, data compression, and encoding by commercial music interests, such as a record company or an entertainment music company, or by an artist. This flow from the submitted music to the compression technology is indicated by the FIG. **13** data flow arrow marked "1". After the compression and encoding, the digital information in the form of music files is uploaded to servers, such as the audio material servers described above. The servers maintain file organization, for example, according to musical genre, artist, album or compilation name, and song title. In addition, the servers may store information pertaining to users, such as usual or preferred selection patterns, so that a predictive loading scheme can be used to direct downloads in of music files. For example, the first ten seconds of a user's one-hundred most frequent selections may be downloaded immediately upon receipt of that user's identification number, to minimize any delay associated with the typical query and response processing. The data flow from the compression technology to the servers is indicated by the FIG. **13** data flow arrow marked "2".

After a user requests a song, the system responds by sending a data stream from a server through any established data transfer line to the music information network. In the preferred embodiment, this network is the Internet. Connecting the users and the servers through the Internet provides a convenient and easily accessible means of transferring the music information from the servers to the users. The data stream includes a copy flag code indicating whether the requested audio material is for immediate listening only or if digital copying and recording rights are also requested. The data flow from the servers to the music information network is indicated by the FIG. **13** data flow arrow marked "3". At the playback unit, indicated as the "Home Audio Component" in FIG. **13**, network interface is used to receive the data stream, as indicated by the data flow arrow marked "4". The data transfer methods may include, for example, use of analog telephone line communication, ISDN services, high-speed cable communications for video and digital data, and the like.

Next, the user identification information provides identification of the listener for billing purposes and for personalization features, such as described above. The user identification information can be entered, if desired, using a card with magnetically encoded user information, so such as a credit card, or the information can be entered manually through the user display interface. A default ID may also be associated with the unit itself. The data flow of personal information from the user to the playback unit is indicated by the FIG. **13** data flow arrow marked "5".

As described above, searching of available audio material may be carried out by a user through the user display interface. The data flow from the user display interface to the playback unit is indicated by the FIG. **13** data flow arrow marked "6" and contains commands issued from the user display interface to the main unit. In the preferred embodiment, the most commonly used controls of the playback unit closely resemble those of a conventional multiple-disc CD player, including disc select, play, fast forward, and the like. These controls, as well as the search functions, can be instantiations of objects that are part of the graphical user interface (GUI) of the user display. This GUI may be replicated on a remote control device, as indicated in FIG. **13**.

After the user has used the GUI to select music, one of the options that can be permitted by the system is to send the digital output of audio material to a storage device for digital recording on storage media, as indicated by the FIG. **13** data flow arrow marked "7". Alternatively, an analog audio signal is sent from the playback unit to audio connections of the home audio system, as indicated by the FIG. **13** data flow arrow marked "8". While music is being delivered to the designated destinations, the user display interface shows information concerning the current and upcoming user selections. The data flow from the playback unit to the user display interface is indicated by the FIG. **13** data flow arrow marked "9".

During the time a user is cleared via the personal information for accessing the servers, the user may upload additional information and queries using the network communications, as indicated by the FIG. **13** data flow arrow marked "10". Using the appropriate network protocols, the correct server is contacted. For example, different servers may have different types of musical genres, or the servers may each store many different types of music. The data flow from the network to the servers is indicated by the FIG. **13** data flow arrow marked "11".

FIG. **14** is a data flow diagram of the playback unit illustrated in FIG. **1**, showing the information that is transmitted among the playback unit components and how those components interact. As noted above in the discussion of FIG. **13**, a network data stream is received at the playback unit. The FIG. **14** data flow arrow marked "1" indicates that the data stream is received at a network interface for the conversion of received data into a data stream that can be used by the playback unit. As noted above, the data may be received through an analog telephone line communication, ISDN services, high-speed cable communications for video and digital data, or similar scheme. Thus, two-way network communication is contemplated for the receipt of this information over the network. Next, the received identification data is provided to the processor of the playback unit. The data flow from the network interface to the processor is indicated by the FIG. **14** data flow arrow marked "2". Initially, the processor sends the received audio material to the memory, as indicated by the FIG. **14** data flow arrow marked "3". The processor then determines how to process and handle the audio material.

In the preferred embodiment, the playback unit includes a digital signal processor (DSP) that is specially designed to perform audio decompression. The processor determines when the received audio material in the form of compressed digital files are sent to the DSP and when the audio material is simply erased or overwritten. This action is represented by the FIG. **14** data flow arrow marked "4". At the DSP, the processed audio material is sent to the digital-to-analog converter (DAC) subsystem for output to the home audio system for listening. The data flow from the DSP to the DAC is indicated by the FIG. **14** data flow arrow marked "5". The data flow from the DAC to the home audio system is indicated by the FIG. **14** data flow arrow marked "6". If digital copying of the audio material has been authorized (such as described above in connection with the data flow arrow of FIG. **13** numbered "7"), then the DSP sends digital output to a digital media storage device. The data flow from the DSP to the storage device is indicated by the FIG. **14** data flow arrow marked "7".

In FIG. **13**, it was noted that a user may upload additional information and queries during the time a user is cleared via the personal information for accessing the servers. It should be understood that such communications must first proceed from the user through the playback unit processor and through the network interface to the servers. In FIG. **14**, the data flow of instruction and information from the user, via

the user display interface, is indicated by the data flow arrow marked "8". The data flow of user information and queries from the processor to the network interface is indicated by the data flow arrow marked "9". The processor may convey return information back to the user through the user display interface, as indicated by the FIG. **14** data flow arrow marked "10".

The information received by the processor includes a copy authorization flag that permits or prohibits digital copying. This permits, if needed, compliance with regulatory schemes such as the U.S.A. Serial Copy Management System (SCMS), or the SDMI format, or forms of digital signature or digital watermark authorizations. Thus, depending on the received information, the processor determines whether the DSP output will be sent to the DAC for output to the home audio system or to a storage device for digital recording. The data flow of processor command to the DSP for output control is indicated by the FIG. **14** data flow arrow marked "11".

The playback unit preferably includes an EEPROM that permits convenient updates of functionality and features. Thus, new programming code and data can be stored into the EEPROM under control of the processor. The data flow of processor command from the processor to the EEPROM is indicated by the data flow arrow marked "12". The data flow of programming instructions and data from the EEPROM to the processor is indicated by the data flow arrow marked "13".

Finally, the audio material, stored in compressed format, may include partial downloads of music files, as commanded by the received data packets (described above). These partial downloads may have been selected, for example, by the predictive downloading schemes that respond to user information (see the data flow arrow of FIG. **13** marked "2"). The partial downloads are received at the playback unit and stored in the memory. The data flow of audio to material from the network interface to the memory is indicated by the FIG. **14** data flow arrow marked "14".

Thus, the music playback system described above provides an interface between a conventional home audio system and a network source for audio material comprising a playback unit with a simple operating system that is not accessed by the user and does not require the launch of special software to initiate playback. Therefore, the playback unit can be operated without special computer skills and without navigating complicated PC-like display windows. Access to audio material and distribution rights are controlled by the DUL servers and audio material servers, thereby enabling the playback unit to retrieve a wide range of digital audio material from the network on demand and vastly expanding the range of music available for playback. In this way, the system permits the reproduction of music using the home audio system, for high quality playback in a comfortable setting, and provides controlled access to audio material and controlled distribution and duplication of the material.

The present invention has been described above in terms of a presently preferred embodiment so that an understanding of the present invention can be conveyed. There are, however, many configurations for network-based music playback systems not specifically described herein but with which the present invention is applicable. The present invention should therefore not be seen as limited to the particular embodiments described herein, but rather, it should be understood that the present invention has wide applicability with respect to network-based music playback systems generally. All modifications, variations, or equivalent arrangements and implementations that are within the scope of the attached claims should therefore be considered within the scope of the invention.

We claim:

1. A playback apparatus for receiving digital audio material from a network server and providing it to a home audio system for playback, the apparatus comprising:

a user interface that receives commands from a user for selection of an audio composition from a network server, for initiating receipt of the digital audio material comprising portions of the selected audio composition, and for controlling playback of the received digital audio material, wherein the user interface includes a display that shows status of the playback;

a memory that contains operating system instructions and that temporarily stores the digital audio material comprising portions of the selected audio composition, such that digital audio material comprising the complete selected audio composition is not available to the user from the memory; and

a processor that executes the operating system instructions stored in the memory to perform apparatus functions in response to the user commands and to initiate the playback of received digital audio material.

2. A playback apparatus as defined in claim **1**, wherein the apparatus functions performed by the processor include display of a menu selection list, from which the user will make the selection of the audio composition.

3. A playback apparatus as defined in claim **2**, wherein the apparatus functions performed by the processor include sending a current song list version for the selected musical category to a network server and receiving an indication from the network server if the current song list is in need of updating.

4. A playback apparatus as defined in claim **2**, wherein the apparatus functions performed by the processor include sending the user selection to a network server and receiving from the network server the network address of audio material comprising the user selection.

5. A playback apparatus as defined in claim **4**, wherein the apparatus sends the user selection to a network directory and user list server and a network audio material server, wherein the directory and user list server checks user information against a list of authorized users to control download of audio material and the audio material server provides the audio material comprising the selected composition.

6. A playback apparatus as defined in claim **5**, wherein the directory and user list server sets a copy authorization flag that is provided to the audio material server, and the audio material server checks the copy authorization flag to determine if digital copying at the apparatus will be permitted.

7. A playback apparatus as defined in claim **6**, wherein the apparatus functions performed by the processor include processing the received audio material to provide an analog so output signal to the home audio system for playback and listening, and processing the received audio material to provide a digital output stream to a storage media only if digital copying is permitted by the copy authorization flag.

8. A playback apparatus as defined in claim **6**, wherein the playback apparatus receives the audio material from the audio material server in a plurality of packets that are temporarily stored in the playback apparatus memory.

9. A playback unit that receives digital audio material from a network server and provides it to a home audio system for playback, the apparatus comprising:

a user interface that receives commands from a user for selection of an audio composition from a network

server, for initiating receipt of the digital audio material comprising portions of the selected audio composition, and for controlling playback of the received digital audio material, wherein the user interface includes a display that shows status of the playback;

a memory that contains operating system instructions and that temporarily stores the digital audio material comprising portions of the selected audio composition, such that digital audio material comprising the complete selected audio composition is not available to the user from the memory; and

a processor that executes the operating system instructions stored in the memory to perform apparatus functions in response to the user commands and to initiate the playback of received digital audio material, wherein the apparatus functions performed by the processor include (1) display of a menu selection list from which the user will make the selection of the audio composition, (2) sending the user selection to a network server and receiving the audio material comprising the user selection, (3) processing the received audio material to provide an analog output signal to the home audio system for playback and listening, and (4) processing the received audio material to provide a digital output stream to a storage media only if digital copying permission was granted by a received copy authorization flag.

10. A playback apparatus as defined in claim **9**, wherein the playback apparatus receives the audio material from the audio material server in a plurality of packets that are temporarily stored in the playback apparatus memory.

11. A playback apparatus as defined in claim **9**, wherein the apparatus functions performed by the processor include sending a current song list version for the selected musical category to a network server and receiving an indication from the network server if the current song list is in need of updating.

12. A playback apparatus as defined in claim **11**, wherein the apparatus sends the user selection to a network directory and user list server and a network audio material server, wherein the directory and user list server checks user information against a list of authorized users to control download of audio material and the audio material server provides the audio material comprising the selected composition.

13. A playback apparatus as defined in claim **12**, wherein the directory and user list server sets a copy authorization flag that is provided to the audio material server, and the audio material server checks the copy authorization flag to determine if digital copying at the apparatus will be permitted.

14. A playback apparatus as defined in claim **13**, wherein the playback apparatus receives the audio material from the audio material server in a plurality of packets that are temporarily stored in the playback apparatus memory.

15. A method of operating a playback apparatus that receives digital audio material from a network server and provides it to a home audio system for playback, comprising the steps of:

selecting an available music category through a user interface supported by an operating system that is stored in memory of the playback apparatus;

sending a current song list version for the selected music category to a network server and receiving an updated song list if the current song list is in need of updating;

selecting an available composition, sending the selected composition title to a network server, and receiving from the network server the network address of a network audio material server at which audio material comprising the user selection is stored;

receiving the audio material from the network audio material server; and

processing the received audio material to provide an analog output signal to the home audio system for playback and listening, and processing the received audio material to provide a digital output stream to a digital storage media only if digital copying is permitted by a copy authorization flag.

16. A method as defined in claim **15**, wherein the step of receiving the audio material comprises receiving the audio material from the network audio material server in a plurality of packets that are temporarily stored in memory of the playback apparatus.

17. A method as defined in claim **15**, further including the steps of sending the user selection to a network directory and user list server and the network audio material server, wherein the directory and user list server checks user information against a list of authorized users to control download of audio material.

18. A method as defined in claim **17**, wherein the copy authorization flag that is checked in the step of processing the received audio material is set by the directory and user list server and is provided to the network audio material server, and the audio material server checks the copy authorization flag to determine if digital copying at the apparatus will be permitted.

19. A method of operating a playback apparatus that receives digital audio material from a network server and provides it to a home audio system for playback, the method comprising:

receiving user commands through an operating system that is stored in semiconductor memory for selection of an audio composition from a network server;

receiving digital audio material from the network server;

temporarily storing the digital audio material comprising portions of the selected audio composition in playback apparatus memory, such that digital audio material comprising the complete selected audio composition is not stored in the memory at the same time; and

processing the received audio material to provide an analog output signal to the home audio system for playback and listening, processing the received audio material to provide a digital output stream to a storage media only if digital copying permission was granted by a received copy authorization flag.

20. A method as defined in claim **19**, wherein the step of receiving digital audio material comprises receiving the audio material from an audio material server in a plurality of packets that are temporarily stored in the playback apparatus memory.

21. A playback apparatus as defined in claim **19**, wherein the apparatus functions performed by the processor include sending a current song list version for the selected musical category to a network server and receiving an indication from the network server if the current song list is in need of updating.

22. A method as defined in claim **21**, wherein the playback apparatus forwards the user request to a network directory and user list server and to a network audio material server,

wherein the directory and user list server checks user information against a list of authorized users to control download of audio material and the audio material server provides the audio material comprising the selected composition upon authorization from the directory and user list server.

23. A method as defined in claim **22**, wherein the directory and user list server sets a copy authorization flag that is provided to the audio material server, and the audio material server checks the copy authorization flag to determine if digital copying at the playback apparatus will be permitted.

24. A method as defined in claim **23**, wherein the playback apparatus receives the audio material from the audio material server in a plurality of packets that are temporarily stored in the playback apparatus memory.

* * * * *

US006487601B1

(12) **United States Patent**
Hubacher et al.

(10) **Patent No.: US 6,487,601 B1**
(45) **Date of Patent: Nov. 26, 2002**

(54) **DYNAMIC MAC ALLOCATION AND CONFIGURATION**

(75) Inventors: **Kenneth Hubacher**, Cedar Park; **Dennis Sposato**, Austin; **Philip C. Theiller**, Pflugerville, all of TX (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/409,592**

(22) Filed: **Sep. 30, 1999**

(51) **Int. Cl.**[7] **G06F 9/445**
(52) **U.S. Cl.** **709/229**; 713/2; 709/222; 709/219
(58) **Field of Search** 709/222, 229, 709/203, 211, 220, 219; 713/2

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,436,905 A		7/1995	Li et al.	370/95.2
5,577,210 A	*	11/1996	Abdous et al.	709/219
5,675,800 A	*	10/1997	Fisher et al.	709/208
5,729,542 A		3/1998	Dupont	370/346
5,974,547 A	*	10/1999	Klimenko	709/217
6,052,779 A	*	4/2000	Jackson et al.	713/2
6,292,890 B1	*	9/2001	Crisan	709/220
6,356,965 B1	*	3/2002	Broyles et al.	709/220

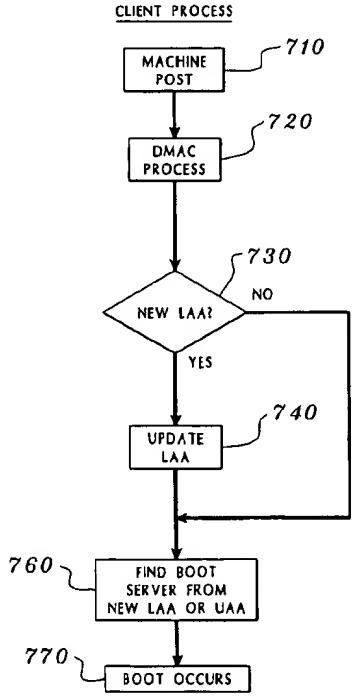
* cited by examiner

Primary Examiner—David Wiley
Assistant Examiner—George Neurauter
(74) *Attorney, Agent, or Firm*—David A. Mims, Jr.; Rudolf O. Siegesmund

(57) **ABSTRACT**

A method and apparatus for Dynamic MAC Allocation and Configuration is based on the ability to remotely boot a client machine from a server machine and adds the capability to assign a Locally Administered Address (LAA) to override the Universally Administered Address (UAA). A set of programs at the workstation allows a remote boot and interaction the server. The client machine will send out a DMAC discovery frame. The discovery frame will be intercepted by a DMAC program installed on the server which will be running and listening for the request. Once the DMAC program intercepts the request it analyzes the request and takes one of two actions. If necessary, the server will run an "initialization" script. For workstations that have already been initialized, the server will send an LAA to the client workstation from a table or pool. The client workstation will then request an operating system with its new LAA. The boot options will be a table or pool corresponding to an LAA or range of LAA's. In order to achieve the override of the UAA, the DMAC will assign an LAA to the workstation. Once the LAA is assigned the boot will proceed based on the package that will be shipped to that address.

6 Claims, 9 Drawing Sheets



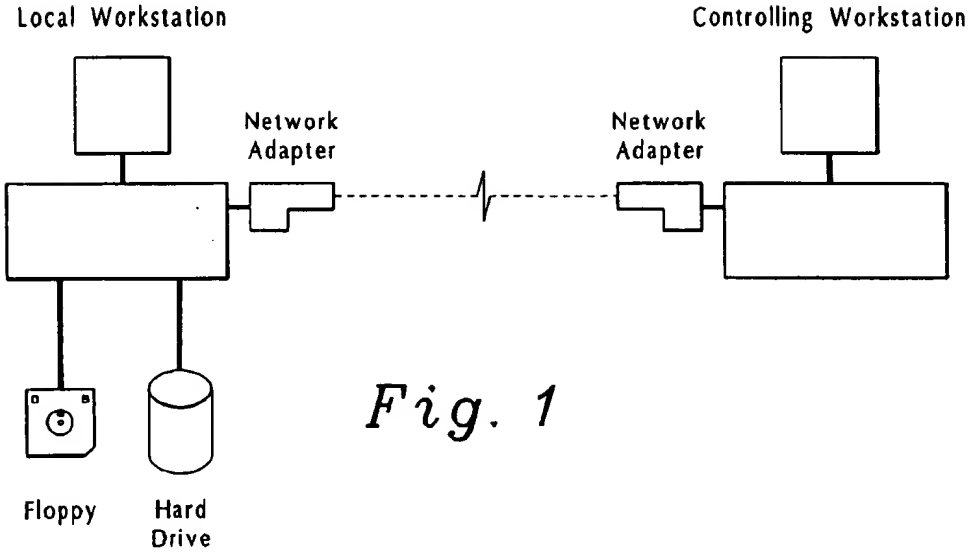

Local Workstation
Controlling Workstation
Network Adapter
Network Adapter
Floppy
Hard Drive

Fig. 1

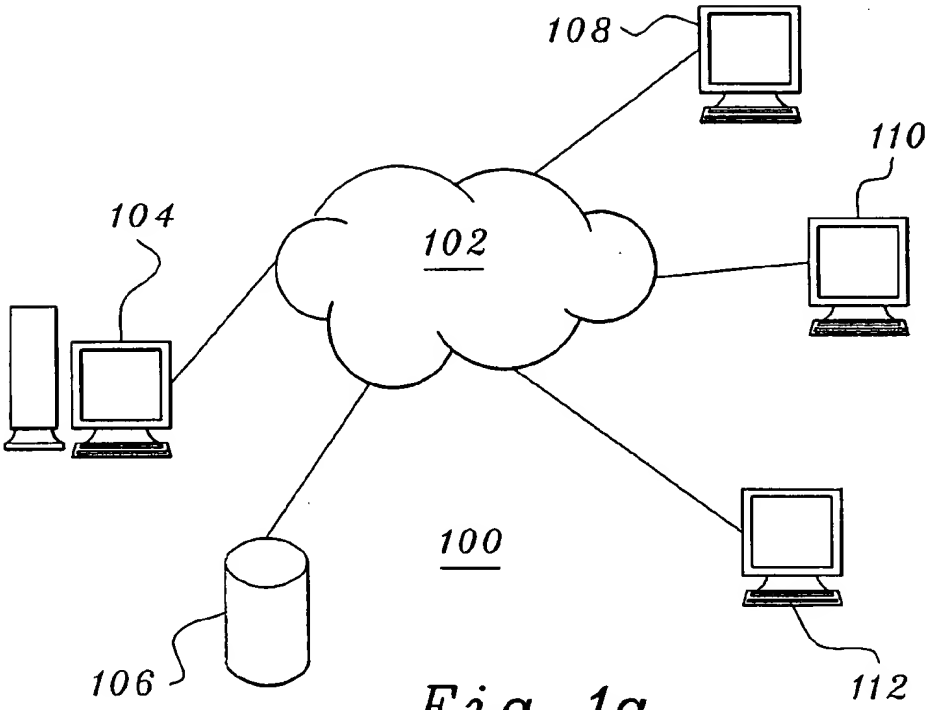

108
110
104
102
100
106
112

Fig. 1a

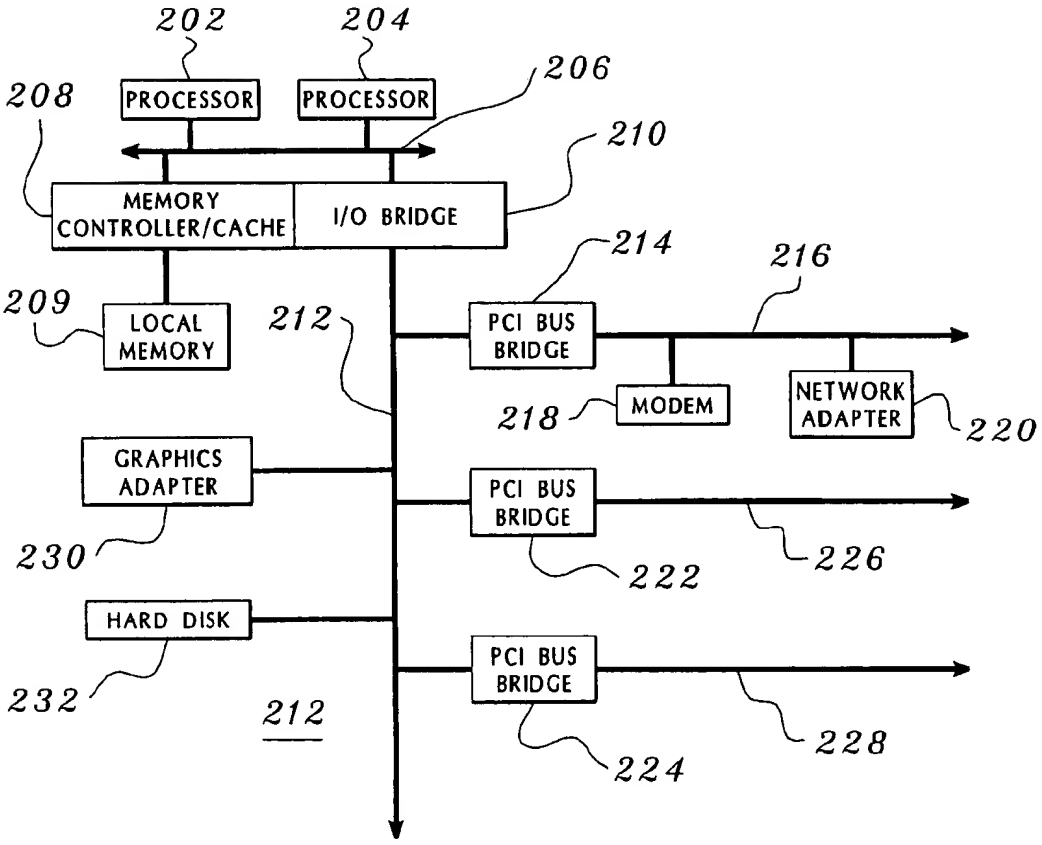

202
204
206
208
PROCESSOR
PROCESSOR
210
MEMORY CONTROLLER/CACHE
I/O BRIDGE
214
216
209
LOCAL MEMORY
212
PCI BUS BRIDGE
218
MODEM
NETWORK ADAPTER
220
GRAPHICS ADAPTER
PCI BUS BRIDGE
230
222
226
HARD DISK
PCI BUS BRIDGE
232
212
224
228

Fig. 2

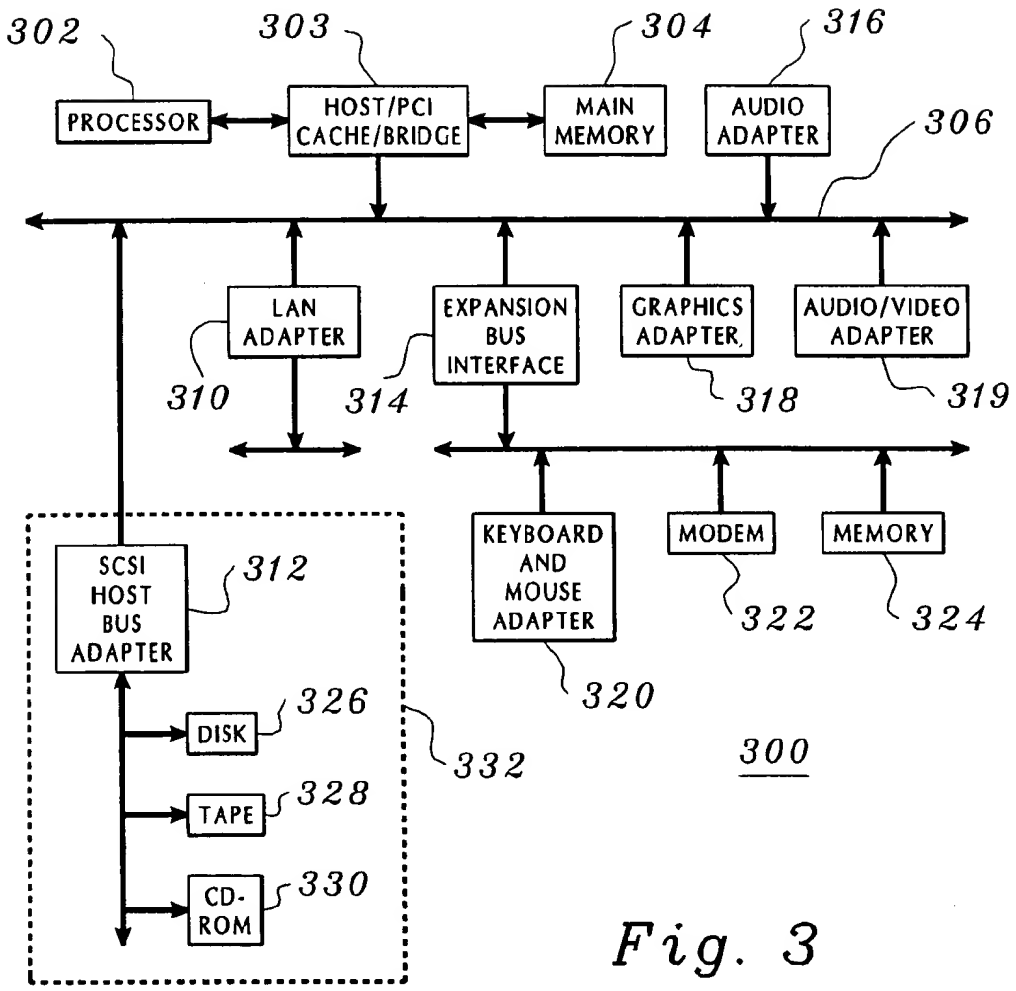

302
303
304
316
PROCESSOR
HOST/PCI CACHE/BRIDGE
MAIN MEMORY
AUDIO ADAPTER
306
LAN ADAPTER
EXPANSION BUS INTERFACE
GRAPHICS ADAPTER
AUDIO/VIDEO ADAPTER
310
314
318
319
SCSI HOST BUS ADAPTER
312
KEYBOARD AND MOUSE ADAPTER
MODEM
MEMORY
322
324
320
326
DISK
332
300
328
TAPE
330
CD-ROM

Fig. 3

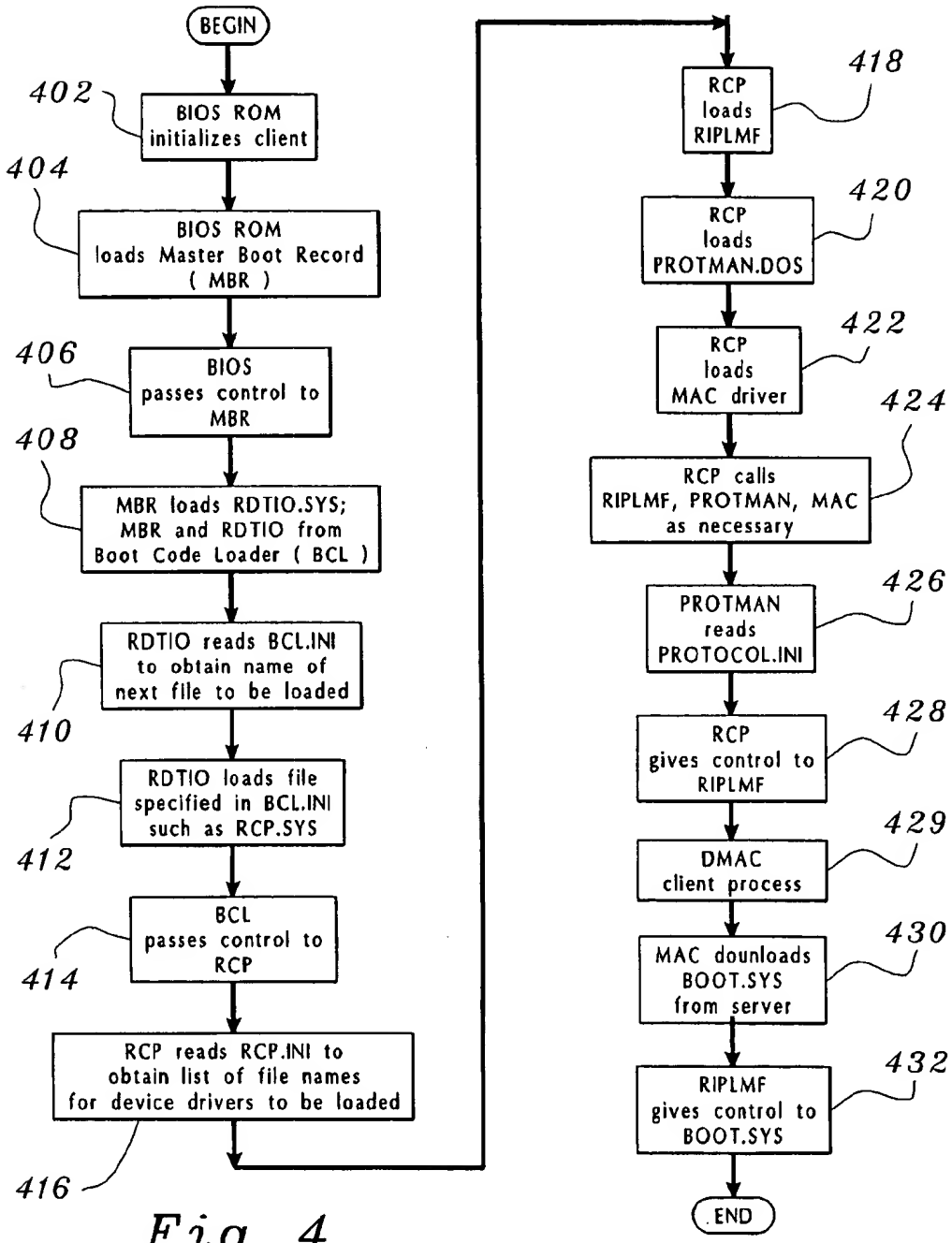

BEGIN
402
BIOS ROM initializes client
404
BIOS ROM loads Master Boot Record (MBR)
406
BIOS passes control to MBR
408
MBR loads RDTIO.SYS; MBR and RDTIO from Boot Code Loader (BCL)
410
RDTIO reads BCL.INI to obtain name of next file to be loaded
412
RDTIO loads file specified in BCL.INI such as RCP.SYS
414
BCL passes control to RCP
416
RCP reads RCP.INI to obtain list of file names for device drivers to be loaded
418
RCP loads RIPLMF
420
RCP loads PROTMAN.DOS
422
RCP loads MAC driver
424
RCP calls RIPLMF, PROTMAN, MAC as necessary
426
PROTMAN reads PROTOCOL.INI
428
RCP gives control to RIPLMF
429
DMAC client process
430
MAC dounloads BOOT.SYS from server
432
RIPLMF gives control to BOOT.SYS
END

Fig. 4

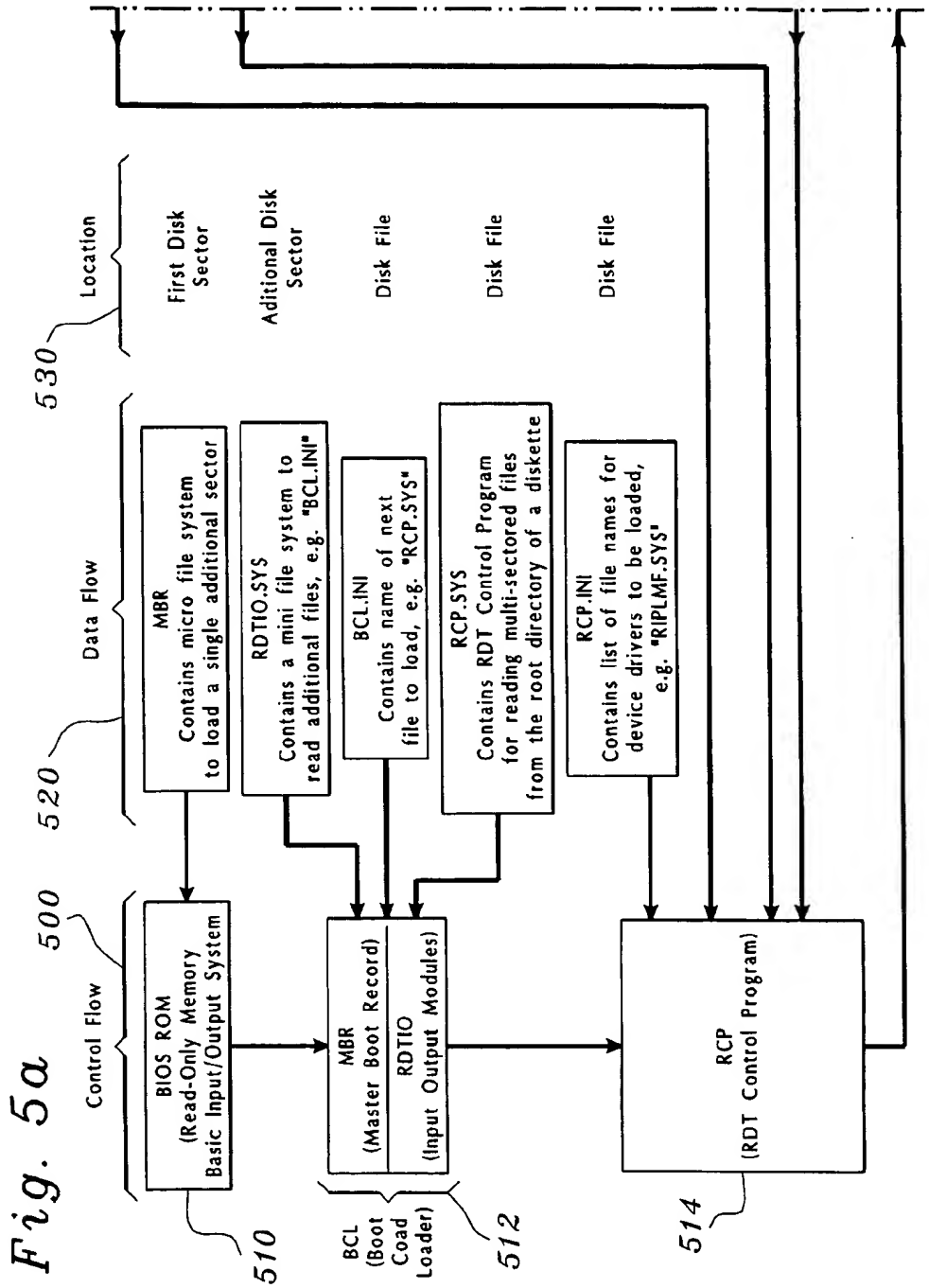

Location
First Disk Sector
Aditional Disk Sector
Disk File
Disk File
Disk File
530
Data Flow
MBR
Contains micro file system to load a single additional sector
RDTIO.SYS
Contains a mini file system to read additional files, e.g. "BCL.INI"
BCL.INI
Contains name of next file to load, e.g. "RCP.SYS"
RCP.SYS
Contains RDT Control Program for reading multi-sectored files from the root directory of a diskette
RCP.INI
Contains list of file names for device drivers to be loaded, e.g. "RIPLMF.SYS"
520
500
Control Flow
BIOS ROM
(Read-Only Memory Basic Input/Output System
MBR
(Master Boot Record)
RDTIO
(Input Output Modules)
RCP
(RDT Control Program)
510
BCL (Boot Coad Loader)
512
514

Fig. 5a

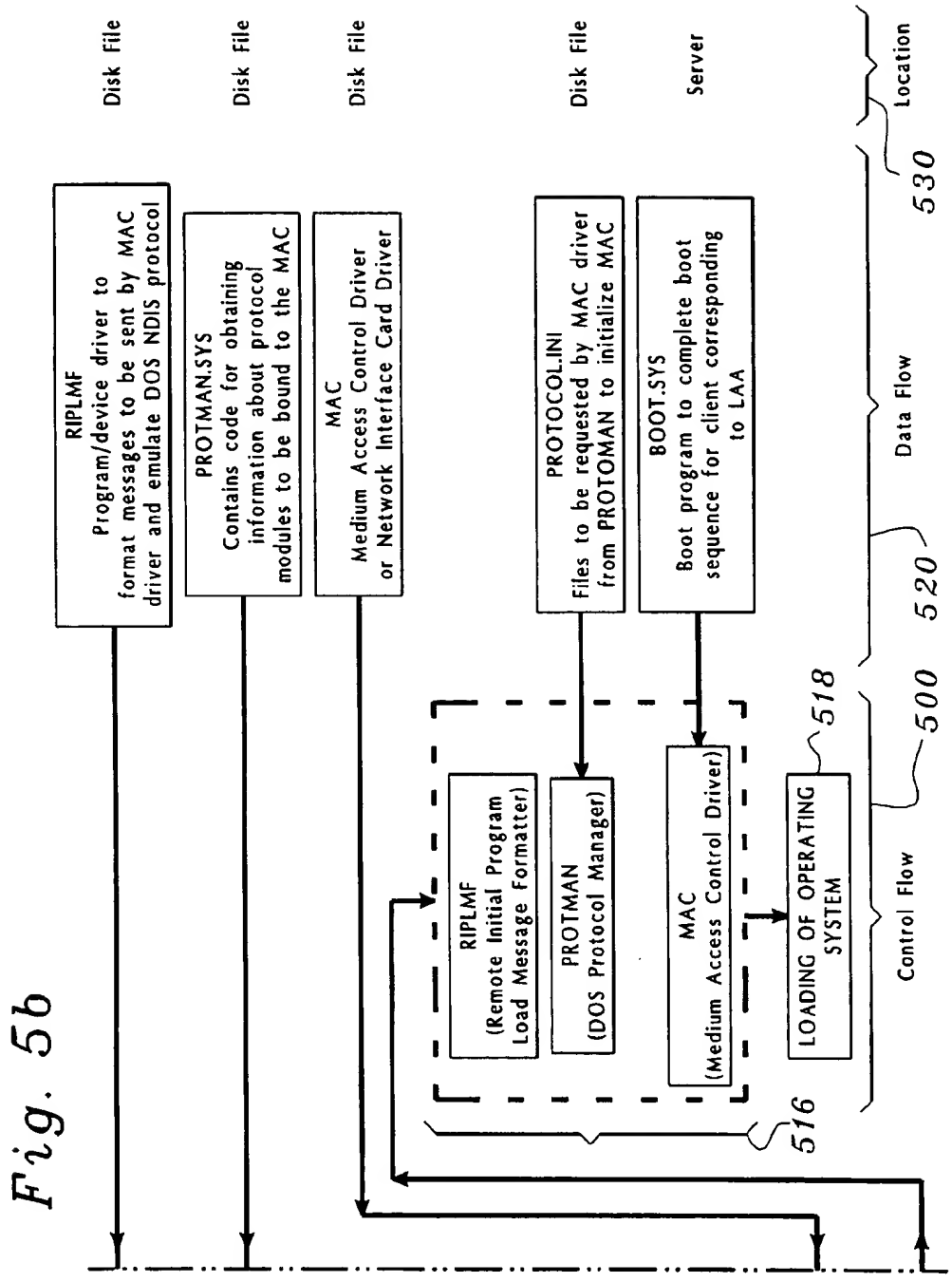

Disk File
Disk File
Disk File
Disk File
Server
Location
RIPLMF
Program/device driver to format messages to be sent by MAC driver and emulate DOS NDIS protocol
PROTMAN.SYS
Contains code for obtaining information about protocol modules to be bound to the MAC
MAC
Medium Access Control Driver or Network Interface Card Driver
PROTOCOL.INI
Files to be requested by MAC driver from PROTOMAN to initialize MAC
BOOT.SYS
Boot program to complete boot sequence for client corresponding to LAA
530
Data Flow
520
500
518
RIPLMF
(Remote Initial Program Load Message Formatter)
PROTMAN
(DOS Protocol Manager)
MAC
(Medium Access Control Driver)
LOADING OF OPERATING SYSTEM
Control Flow
516

Fig. 5b

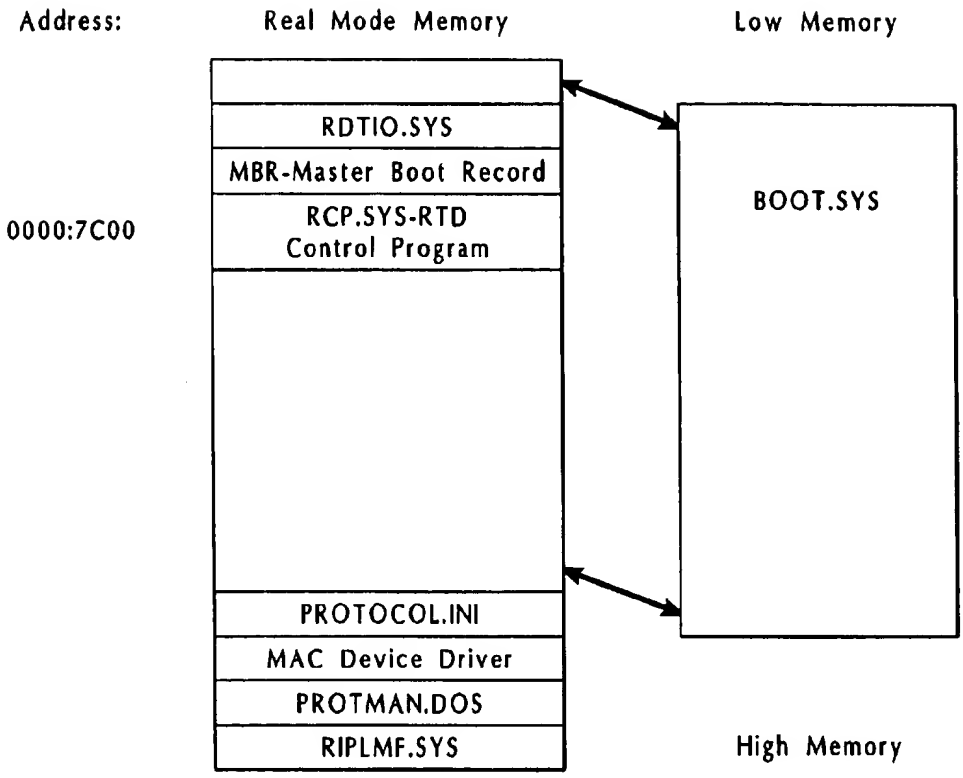

Address:
Real Mode Memory
Low Memory
RDTIO.SYS
MBR-Master Boot Record
0000:7C00
RCP.SYS-RTD
Control Program
BOOT.SYS
PROTOCOL.INI
MAC Device Driver
PROTMAN.DOS
RIPLMF.SYS
High Memory

Fig. 6

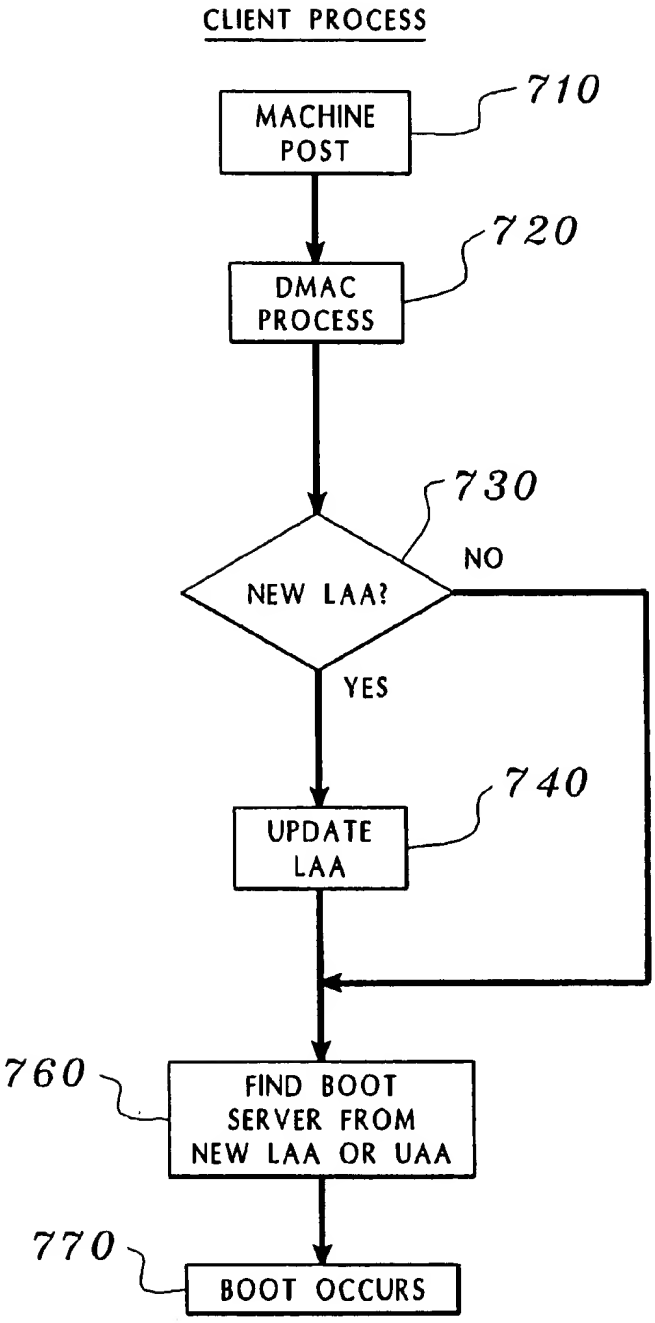

CLIENT PROCESS
MACHINE POST
710
DMAC PROCESS
720
730
NEW LAA?
NO
YES
UPDATE LAA
740
760
FIND BOOT SERVER FROM NEW LAA OR UAA
770
BOOT OCCURS

Fig. 7

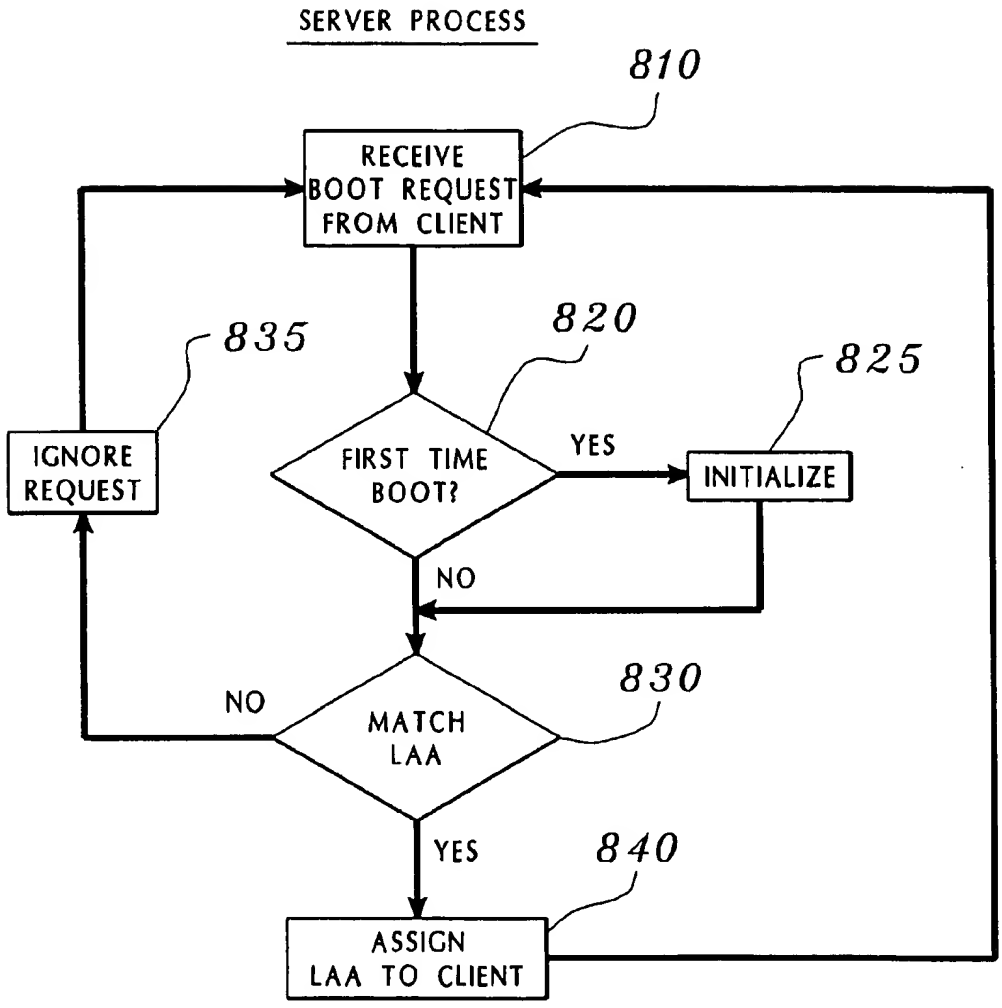

SERVER PROCESS
810
RECEIVE BOOT REQUEST FROM CLIENT
820
FIRST TIME BOOT?
YES
825
INITIALIZE
NO
830
MATCH LAA
NO
835
IGNORE REQUEST
YES
840
ASSIGN LAA TO CLIENT

Fig. 8

DYNAMIC MAC ALLOCATION AND CONFIGURATION

FIELD OF THE INVENTION

The present invention relates to the control of a client computer system's Medium Access Control (MAC) address by a server computer system.

BACKGROUND OF THE INVENTION

A computer or computer system, when turned on, must be prepared for operation by loading an operating system. In the normal operation of a single computer system, when a user issues a boot command to the computer, the computer responds to the boot command by attempting to retrieve the operating system files from the computer systems memory. Configuration data files are also needed to configure the specific machine with the hardware parameters necessary for the specific hardware configuration. These files also contain information needed to initialize videos, printers, and peripherals associated with the particular machine. For example, the files would include CONFIG.SYS in the MS-DOS operating system, available from Microsoft Corporation.

Computers or computer systems can be connected in a network normally consisting of a client workstation, a server and a central network. In a system where the computer's storage is maintained when the power is turned off, the operating system can be stored in the computer itself. In a system where the computer has only storage that is lost when the power is turned off, the computer cannot retrieve the boot information from within the computer itself. In that case, the client sends a request for the operating system files via the network to the server acting as a boot server. Even when the client workstation has non-volatile storage capability, it is advantageous to boot from the server because memory space is saved in the workstation computer. As operating system and application programs expand to provide new and greater capabilities, booting from a server can be highly advantageous.

Several methods of remote booting exist in the marketplace. One is called Remote Initial Program Load (RIPL). RIPL is the process of loading an operating system onto a workstation from a remote location. The RIPL protocol was co-developed by 3Com, Microsoft, and IBM. It is used today with IBM OS/2 Warp Server, DEC Pathworks, and Windows NT. Two other commonly used Remote IPL protocols are a Novell NCP (NetWare Core Protocol), and BOOT-P, an IEEE standard, used with UNIX and TCP/IP networks.

RIPL is achieved using a combination of hardware and software. The requesting device, called the requester or workstation, starts up by asking the loading device to send it a bootstrap program. The loading device is another computer that has a hard disk and is called the RIPL server or file server. The RIPL server uses a loader program to send the bootstrap program to the workstation. Once the workstation receives the bootstrap program, it is then equipped to request an operating system, which in turn can request and use application programs. The software implementations differ between vendors, but theoretically, they all perform similar functions and go through a similar process. The client workstation requires a special Read Only Memory (ROM) installed on its (Local Area Network) LAN adapter or Network Interface Card (NIC). The special ROM is known generally as a remote boot ROM, but two specific examples of remote boot chips are the RIPL chip, which supports ANSI/IEEE standard 802.2, and the Preboot Execution

Environment (PXE) chip, which is used in the Transmission Control Protocol/Internet Protocol (TCP/IP) environment.

While the process has many advantages for booting a computer that has volatile storage, such as a network computer, the computer is required to have a remote boot ROM on the LAN adapter or Network Interface Card (NIC). The remote boot ROM requirement does not allow any user interaction with the remote boot process.

Application Ser. No. 09/329,457 discloses a remotely controlled boot process allowing a client computer to boot from a server without the remote boot ROM requirement.

The client's Medium Access Control (MAC) address is the key factor that determines many characteristics of the boot process. The MAC address determines what server the client will boot from, what operating system will be loaded and what the client's computers configuration will be.

However, in the server-managed client environment, there currently does not exist a way to automatically assign and configure a client's MAC address. The MAC address can be a Universally Administered Address or a Locally Administered Address. The Universally Administered Address (UAA), in a local area network, is the address permanently encoded in an adapter at the time of manufacture. All Universally Administered Addresses are unique. A Locally Administered Address (LAA), in a local area network, is an adapter address that the user can assign to override the Universally Administered Address. Therefore, a need exists for an automatic way of configuring and distributing a Locally Administered Address (LAA). If this can be done, then when the boot configuration is changed, an address corresponding to the configuration desired can be assigned. Such a system would provide a seamless solution in dynamically changing a client's boot environment and would greatly expand the ability of the administrator to remotely configure the client machines within a network.

SUMMARY OF THE INVENTION

The invention meeting the needs identified above is a method and apparatus for Dynamic MAC Allocation and Configuration. Such a system is based on the ability to remotely boot a client machine from a server machine and adds the capability to assign a Locally Administered Address (LAA) to override the Universally Administered Address (UAA).

The first part of the process is to set up the capability for remote booting. In the preferred embodiment, a set of programs at the workstation allows a remote boot and interaction with a program on the server. Instructions from a Basic Input Output System (BIOS) ROM are executed to load a Boot Code Loader (BCL) from a nonvolatile, read/write memory, such as a diskette or hard disk. The BCL executes to load a Remote Control Program (RCP), and the RCP executes to load a message program, a protocol manager and/or device drivers without loading an operating system. The message program and/or device drivers communicate with a Dynamic Mac Allocation and Configuration (DMAC) program in the network server. First, the program will interface with an NDIS compliant Network Interface Card (NIC) to send out a DMAC discovery frame. At this point the workstation seeks MAC specific information. The discovery frame will be intercepted by a DMAC program installed on the server which will be running and listening for the request. Once the DMAC program intercepts the request it will analyze the request and take one of two actions. First, if this is the first time that the client machine has been booted, the server will run an "initialization" script.

In other words the DMAC will prepare the other boot servers by informing them that in the future, the workstation in question will boot. The workstation will be placed in a MAC table or pool. The second action, for workstations that have already been initialized, is that the server, based on the information received, will send an LAA to the client workstation from the table or pool. The client workstation will then request an operating system with its new LAA. The boot options will be a table or pool corresponding to an LAA or range of LAA's. In other words, a particular boot option or package will be sent to a system making a request that has the corresponding LAA. In order to achieve the override of the UAA, the DMAC will assign an LAA to the workstation. Once the LAA is assigned the boot will proceed based on the package that will be shipped to that address.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

FIG. **1** depicts an overview of the system.

FIG. 1A depicts a distributed data processing system.

FIG. **2** depicts a block diagram of a server.

FIG. **3** depicts a block diagram of a work station.

FIG. **4** depicts a flow chart of the workstation process.

FIG. **5** depicts a flow chart of the workstation process.

FIG. **6** depicts a diagram of workstation memory.

FIG. **7** depicts a flow chart of the workstation process.

FIG. **8** depicts a flow chart of the server process.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 depicts a pictorial representation of a distributed data processing system in which the present invention may be implemented and is intended as an example, and not as an architectural limitation, for the processes of the present invention. Distributed data processing system **100** is a network of computers which contains a network **102**, which is the medium used to provide communications links between various devices and computers connected together within distributed data processing system **100**. Network **102** may include permanent connections, such as wire or fiber optic cables, or temporary connections made through telephone connections. In the depicted example, a server **104** is connected to network **102** along with storage unit **106**. In addition, clients **108**, **110**, and **112** also are connected to a network **102**. Clients **108**, **110**, and **112** may be, for example, personal computers or network computers.

For purposes of this application, a network computer is any computer, coupled to a network, which receives a program or other application from another computer coupled to the network. In the depicted example, server **104** provides data, such as boot files, operating system images, and applications to clients **108**, **110** and **112**. Clients **108**, **110**, and **112** are clients to server **104**. Server **104** may also act as a boot server because it stores the files and parameters needed for booting each of the unique client computers systems **108**, **110**, and **112**. Distributed data processing system **100** may include additional servers, clients, and other devices not shown. In the depicted example, distributed data processing system **100** is the Internet with network **102** representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. Distributed data processing system **100** may also be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). Referring to FIG. **2**, a block diagram depicts a data processing system, which may be implemented as a server, such as server **104** in FIG. **1** in accordance with the present invention. Data processing system **200** may be a symmetric multiprocessor (SMP) system including a plurality of processors **202** and **204** connected to system bus **206**. Alternatively, a single processor system may be employed. Also connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**. I/O bus bridge **210** is connected to system bus **206** and provides an interface to I/O bus **212**. Memory controller/cache **208** and I/O bus bridge **210** may be integrated as depicted Peripheral component interconnect (PCI) bus bridge **214** connected to I/O bus **212** provides an interface to PCI local bus **216**. Modem **218** may be connected to PCT bus **216**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers **108**, **110** and **112** in FIG. 1 may be provided through modem **218** and network adapter **220** connected to PCI local bus **216** through add-in boards. Additional PCI bus bridges **222** and **224** provide interfaces for additional PCI buses **226** and **228**, from which additional modems or network adapters may be supported. In this manner, server **200** allows connections to multiple network computers. A memory-mapped graphics adapter **230** and hard disk **232** may also be connected to I/O bus **212** as depicted, either directly or indirectly. Those of ordinary skill in the art will appreciate that the hardware depicted in FIG. **2** may vary. For example, other peripheral devices, such as optical disk drive and the like also may be used in addition or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention. The data processing system depicted in FIG. **2** may be, for example, an IBM RISC/System 6000 system, a product of International Business Machines Corporation in Armonk, N.Y., running the Advanced Interactive Executive (AIX) operating system.

With reference now to FIG. **3**, a block diagram illustrates a data processing system in which the RCP may be implemented. Data processing system **300** is an example of either a stand-alone computer, if not connected to distributed data processing system **100**, or a client computer, if connected to distributed data processing system **100**. Data processing system **300** employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Micro Channel and ISA may be used. Processor **302** and main memory **304** are connected to PCI local bus **306** through PCI bridge **308**. PCI bridge **308** also may include an integrated memory controller and cache memory for Processor **302**. Additional connections to PCI local bus **306** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **310**, SCSI host bus adapter **312**, and expansion bus interface **314** are connected to PCI local bus **306** by direct component connection. In contrast, audio adapter **316**, graphics adapter **318**, and audio/video adapter (A/V) **319** are connected to PCI local bus **306** by add-in boards inserted into expansion slots. Expansion bus interface **314** provides a connection for a keyboard and mouse adapter **320**, modem

322, and additional memory 324. SCSI host bus adapter 312 provides a connection for hard disk drive 326, tape drive 328, and CD-ROM 330 in the depicted example. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors. An operating system runs on processor 302 and is used to coordinate and provide control of various components within data processing system 300 in FIG. 3. The operating system may be a commercially available operating system such as OS/2, which is available from International Business Machines Corporation. "OS/2" is a trademark of International Business Machines Corporation. An object oriented programming system, such as Java, may run in conjunction with the operating system and provides calls to the operating system from Java programs or applications executing on data processing system 300. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs may be located on storage devices, such as hard disk drive 326, and they may be loaded into main memory 304 for execution by processor 302. Those of ordinary skill in the art will appreciate that the hardware in FIG. 3 may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in FIG. 3. Also, the processes of the present invention may be applied to a multiprocessor data processing system. For example, data processing system 300, if optionally configured as a network computer, may not include SCSI host bus adapter 312, hard disk drive 326, tape drive 328, and CD-ROM 330, as noted by the box with the dotted line in FIG. 3 denoting optional inclusion. In that case, the computer, to be properly called a client computer, must include some type of network communication interface, such as LAN adapter 310, modem 322, or the like. As another example, data processing system 300 may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system 300 comprises some type of network communication interface. As a further example, data processing system 300 may be a Personal Digital Assistant (PDA) device which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data. The depicted example in FIG. 3 and above-described examples are not meant to imply architectural limitations with respect to the present invention. It s important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in a form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such a floppy disc, a hard disk drive, a RAM, and CD-ROMs, and transmission-type media, such as digital and analog communications links.

With reference now to FIG. 4, a flowchart depicts the steps used in the DMAC. When a PC is booted, the BIOS ROM chip initializes the system by executing POST (Power-on Self-Test) code and by setting up the BIOS vector tables in low memory and by selecting a boot source (step 402). On newer systems, this is a selectable parameter in the hardware system BIOS setup procedure. If the system has been allowed to select a diskette drive as a boot device, BIOS ROM instructions reads the first sector of the diskette into memory at a predefined location (step 404). This sector is called the Master Boot Record (MBR). The BIOS then gives control to MBR (step 406).

The MBR, Cyl 0, Head 0, Sector 1 of the diskette is the first thing loaded into the client system after POST processing is completed. The MBR module is no larger than 512 bytes and contains the partition boot table for the diskette. The MBR module also contains an identity stamp and a micro file system with the ability to load a RDT Input Output Module (RDTIO) (step 408). The micro file system has the capability of reading one sector of any file contained in the root directory of a diskette employing the 12-bit FAT architecture. The file name for RDTIO is hard-coded in the MBR and is not user-changeable.

RDTIO.SYS is a one sector file that is read into memory by the MBR. Together, the RDTIO and the MBR make up the BCL. RDTIO adds enough capability to the file system to allow reading additional files.

The first file read by BCL is its initialization file. This file, BCL.INI, contains the name of a self loading, multi-sectored file that can be found in the root directory of the diskette. When this file is successfully read into memory, it will be give control and BCL will no longer be required. The syntax for the BCL.INI file is very restrictive. There are only two parameters in capital letters. The parameters are the name of the file to load and the address of the desired location. For example, RCP.SYS,0000:7C00. This instructs the BCL to load file RCP.SYS at location 000:7C00 in real memory. There is no error checking so the file must be in the root directory of the diskette. If the address field is optional, the 7C00 is the default. However, if the address filed is used, the address filed allows the BCL to load images directly into memory from the disk or diskette. The file name must start at the first location in the file.

BCL.INI contains the name of the next file to load (step 410). BCL.INI specifies a file that contains a self-supporting program or module, as it will be given control at a predetermined area and BCL will terminate execution, leaving the newly loaded module on its own. In RCB's case, this file's name is RCP.SYS-RDT, or the "RDT Control Program" which consists of RCP.SYS and RCP.INI. The RDT Control Program is loaded by the BCL. Once loaded, there is no longer any dependency on the BCL for services. The RCP contains its own mini file system consisting of enough logic to read multi-sectored files from the root directory of a diskette which is formatted using 12-bit FAT architecture.

RDTIO loads RCP.SYS (step 412) and passes control to RCP (step 414). RCP's task is to load additional files, such as device drivers, provide DOS function emulation in support of these drivers, and load other components of RCB, specifically the RIPL Message Formatter (RIPLMF). RCP receives its instructions from an .INI file called RCP.INI (step 416) in the root directory of the diskette. These instructions are in the form of file names. RCP.INI will be parsed and displayed on the console as it is used. The purpose of the INI file is to tell RCP which drivers it needs to load to support the particular NIC on the system in which it is running. The RCP also has the responsibility of providing DOS function emulation to the device drivers when they are in their initialization routines. RCP allows the device drivers to execute as though they were in a real DOS environment. RCP further allows different drivers to be loaded for individual NICs without forcing source code changes in the RCP. The syntax is:

root of disk RCP loads

msgf=[file name] where "file name" is the name of any "Message Formatter" that is to be used for this boot, i.e.: "msgf=riplmf.sys".

load=[file name] where "file name" is the name of any module that has to be loaded to make RCB work. At a minimum, the DOS device drivers, for the NIC in the machine, must be identified this way, i.e.: "load=device.sys".

"ip=[ip address] where "ip address" is the dotted decimal IP address, i.e.: ip=123.456.789.012."

mac=[mac address] where "mac address" is the 12 hex digit MAC address in a continuous string, i.e.: "mac=001122334455".

Each entry must be separated by any, or all, of the following characters:

20h=Space

ØAh=Carriage return

ØDh=line feed

Almost any editor can be used to create the file.

The RIPLMF is loaded first (step **418**) followed by the device drivers. The DOS Protocol Manager (PROTMAN.DOS) is usually loaded next (step **420**) followed by the NIC driver, also referred to as the MAC driver (step **422**). The RCP will call each driver (step **424**), in turn, allowing it to perform its initialization routines, open files, display messages, etc. PROTMAN.DOS will request a file called PROTOCOL.INI to be read in during this time (step **426**). This file is requested by the MAC driver from PROTMAN during an inter-module conversation when the MAC is initialized. The MAC causes messages to be sent and received on the LAN.

PROTMAN.DOS is the DOS protocol manager device driver. According to the NDIS specification, "the Protocol Manager reads the PROTOCOL.INI file at INIT time and parses it to create the configuration memory image passed to the protocol modules." The RCB uses it for just that purpose. The MAC driver will issue Input/Output Controls (IOCTLs) to PROTMAN to get this information, as well as information about the protocol drivers that wish to be bound to it. RIPLMF presents itself to PROTMAN.DOS as though it were a protocol driver requesting to be bound to the MAC. This is done by placing entries in the PROTOCOL.INI file which make RIPLMF look like a protocol driver and through IOCTL calls from RIPLMF to PROTMAN.DOS. RCB emulates most of the other additional BindAndStart and InitiateBind logic which, in a DOS environment, comes from additional support programs. These programs are unnecessary in the RCB system.

The PROTOCOL.INI file used by RCB can be the same one that is included in the BOOT.SYS image assembled in the server with some minor changes. The MF has to be added to it as follows:

```
[RIPLMF-MOD]
D riverName=RIPLMF$
Bindings=ELPC3
```

The "Bindings=" statement must point to the MAC driver, in this case ELPC3. The example above was taken from the PROTOCOL.INI used with the 3Com 3C589 PCMCIA Ethernet card.

The entire file looks like this:

```
[protman$]
   Driver name=protman$
[ELPC3]
   Driver name=ELPC3$
   PCMCIA_ENABLER=YES
[RIPLMF-MOD]
   Driver name=RIPLMF$
   Bindings=ELPC3
```

The device drivers used by RCB are also called ANSI/IEEE standard 802.2 drivers. RCB requires the drivers specific to the DOS environment. The EL90X.DOS is used to support the #Com3C509 PCI Ethernet card. The ELPC3.DOS driver supports the 3Com#C589 PCMCIA Ethernet card.

When all initialization is complete, RIPLMF is given control (step **428**), and the services of RCP are no longer required. RIPLMF is a hybrid application program and NDIS protocol device driver. It follows the NDIS specification in its actions with both PROTMAN and the MAC driver. RIPLMF's relationship to these two other programs is that of a protocol driver; however, RIPLMF also "formats" messages and present them to the MAC for delivery. Since the other drivers must be made to believe they are working in an NDIS environment, RIPLMF also does emulation in two areas, "BindAndStart" and "InitiateBind." According to NDIS, a protocol driver must be bound to a MAC driver. Therefore, RIPLMF binds to the MAC such that the MAC cannot tell the difference between RIPLMF or a DOS NDIS protocol driver.

At this point the client will send out a DMAC discovery frame through RIPLMF. The discovery frame will be sent out and the client machine will wait for any specified time out period. If there is a server responsive to the frame, the server will send back an LAA. Upon receipt of the LAA, the original MAC address is overridden and the client appears to any server as the LAA just assigned.

Once the LAA has been assigned, the RIPLMF asks the MAC to communicate with the server to obtain the boot files. RIPLMF asks the MAC to send: "Find" and "GetFile. The find message is replied to by a "Found" from the server. Once RIPLMF knows the server has been found, it sends out the getfile message. The server responds by sending the boot package to the client which corresponds to the LAA designated by the administrator.

When all segments of programs assigned by the administrator have been received, RIPLMF resets any vectors that may have been used by RCP and the other drivers, and gives the system over to the programs sent to Boot.sys corresponding to the LAA sent by the server. At this point no components of RCP are required, nor can they be found in the system. The find/found dialog is based on the LAA received from DMAC. The administrator will have made the decisions about which choices will be sent to which workstations.

When this file is downloaded (step **430**), RIPLMF will perform some housekeeping routines and give control to Boot.sys (step **432**). Boot.sys then completes the boot process to load an operating system from the network server (step **434**) based on the LAA assigned in the table by the administrator. DMAC is the controlling mechanism that interacts with each and every client request for an LAA. The network interface between the client and DMAC may be IP based which means that the machine the DMAC is running on must also be running TCP/IP. DMAC receives UDP datagram request from the client and sends back the information in a UDP packet. One implementation would be written in JAVA. DMAC could be run on any platform that has a JVM and is TCP/IP enabled. The following languages are suitable for the programs Assembler, C, C++, Cobol, Pascal, Java, SmallTalk, Perl, Rexx, LISP, APL, BASIC, PLI, PLII. The following protocols are suitable: NETBIO; TCP/IP; 802.2; SNA, SNB, IPX and APPLETALK.

Everything that occurs in the workstation computer is based on the MAC address, which is a hardware name embedded in the chip. Another name for the MAC address is the UAA. Therefore, if the UAA can be overridden and a new number assigned, the package sent to the address can be controlled remotely and automatically. A server on the LAN that recognizes the client computer's MAC address will respond in a pre-determined way. The DMAC allows the assignment of pre-selected LAA's that can provide different boots for different uses.

The RPL/PXE Emulation of the first programs further allows the option of remote booting of multiple operating systems. For example, with the RPL/PXE emulation and its ability to alias the MAC address, the DMAC can offer different operating systems from the same server, different operating systems from different servers, different versions of the same operating system from the same server and different versions of the same operating system from different servers. Moreover, DMAC can be offered from a primary server, a backup server or a different server. Additionally, DMAC can present different workstation functions.

To implement these types of options the administrator would define the appropriate LAA's in a table assigning the LAA's to specific operating systems or packages of operating systems, drivers and applications. The when a request is received from workstation the LAA corresponding to the pre-selected package assigned by the administrator can be sent and assigned to the workstation. The DMAC would then follow through by sending the appropriate operating system or package to the now assigned LAA. For example, for an administrator to give workstations the ability to automatically boot and/or boot and receive applications, the administrator would define an LAA corresponding to the operating system or package the administrator wanted to be automatically sent to that workstation. DMAC would, upon receipt of the UAA for that workstation, override the UAA with the LAA of the desired package and then the package would be sent to the LAA. Only the server where that particular LAA address is defined will respond.

With reference now to FIG. **5**, a flowchart depicts the control flow, the data flow, and the location of data and instructions used in the Dynamic MAC Allocation and Configuration. This figure provides a slightly different perspective compared with FIG. **4**, showing the manner in which files are loaded and then the order in which the code segments within the files obtain control. Control flow **500** shows the manner in which a program, device driver, or set of instructions passes control from one component to another. A generalized sequence of steps performs part of the boot sequence of the client, and each step completes a portion of the sequence before relinquishing control to the next portion. Each of these components comprises instructions that are executed to perform a set of functions. BIOS ROM **510** initializes the client, loads BCL **512**, and passes control to BCL **512**. As shown, BCL **512** may contain a plurality of components that are not necessarily executed sequentially before relinquishing control. Once BCL **512** has loaded RCP **514**, BCL **512** passes control to RCP **514**, which loads components **516**, which may contain programs and/or device drivers. RCP **514** may direct control of components **516** or may pass control to components **516**, which are not necessarily executed sequentially. Once operating system **518** has been retrieved from the server, control of the client computer is relinquished to operating system **518**, which then proceeds to complete the boot process for the client. Data flow **520** shows the data or set of instructions which are loaded by the software components shown as control flow **510**. Although the components in data flow **520** have been given names, these file names may be used for representative purposes only. Other configurations of components in data flow **520** may also be incorporated, and the depicted example in FIG. **5** is not meant to imply configural limitations with respect to the present invention. Locations **530** provide information on the source location for the components in data flow **520**.

With reference now to FIG. **6**, a block diagram depicts a memory map of real mode memory in a 80x86 machine as used in the present invention. Virtually all PC's in use today allow real mode addressing from location zero (0000:0000) to 640 k (AOOO:0000). The diagram shows that the BCL, consisting of the MBR and RDTIO, locate themselves in low storage, and load RCP.SYS at location 0000:7COO. This is actually a predefined location where code will be loaded by the BIOS when booting from a diskette. RCP then loads all required modules into the highest addresses possible. This is done so that the boot blocks for the operating systems to be sent by the server can be loaded in low memory at the operating systems own requested location. When all drivers have been loaded and initialized, RCP gives control to RIPLMF in high memory and is no longer required. RIPLMF will load Boot.Sys for the operating system corresponding to the assigned LAA over all of RCB's code in low memory. This can be done because all DOS emulation, which was done by the RCP, is no longer required. RIPLMF acts as both an application program and NDIS protocol device driver. As such, there is a guarantee that DOS emulation will not be necessary.

FIG. **7** depicts the process at the workstation. The first step is the Machine Power On Self Test (POST) (**810**). The Machine is powered on and goes through its standard power on testing before giving control to the boot manager process. Next, the DMAC process attempts to communicate with the controlling server for the LAA. If contact is made with the controlling server it will result in the receipt of the LAA (**730**). If contact cannot be made with the server, the process will proceed to step **760** to find a boot server based on the old UAA. If a new LAA is assigned, then the LAA will override the old UAA (MAC address) (**740**). The client will ask for the boot server with the new LAA (**760**). Boot will proceed based on the new LAA (**770**).

FIG. **8** depicts the process at the server. First the DMAC receives the MAC address also known a the UAA from the workstation (**810**). The DMAC determines if this a first time boot for that UAA (**820**). If it is, then DMAC will run an initialization routine (**825**). The purpose of the initialization script would be to inform all of the servers in the network that the workstation computer is in the system and will be booting in the future. Second, if the workstation has been previously booted, DMAC will analyze the frame and query specific servers that can handle the boot request. DMAC then sends a specific LAA to the workstation. The LAA will have been assigned by the administrator. The administrator can assign LAA rigidly in a table or flexibly in a pool. If the administrator uses a pool incoming UAA's in a particular range will be assigned a LAA in a range selected by the administrator. If it is not a first time boot, DMAC will seek to matches the UAA against the LAA's or ranges of LAA's on file in the server (**830**). The UAA address for the workstation must correspond to an LAA or range of LAA's chosen by the administrator. If no match is made the request is ignored (**835**). If a match is made then the LAA is transmitted to the workstation and the client process proceeds as in FIG. **7** (**840**).

The advantages provided by the present invention should be apparent in light of the detailed description provided

above. The description of the present invention has been presented for purposes of illustration and description, but is not limited to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention the practical application and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed:

1. A method for booting one or more workstation computers from one or more server computers comprising the steps of:

sending a first request for a Locally Administered Address from the workstation to a server;

receiving the Locally Administered Address from the server at the workstation;

sending a second request for at least one program from the server;

receiving at least one program addressed to the Locally Administered Address from the server in response to said second request;

booting said workstation from said program.

2. A programmable apparatus for presenting pre-selected choices for booting a workstation to a user of the workstation comprising,

programmable hardware comprising;

at least one server computer; and

a plurality of workstation computers;

a plurality of network interface cards connected to said programmable hardware;

a network connecting said server computer and said workstation computers;

a first program installed on said workstation computers;

a second program installed on said server computer for assigning one or more Locally Administered Addresses in response to one or more requests from one or more of the workstation computers;

a plurality of operating systems installed on said server computers;

wherein at least one of said workstation computers is directed by said first program to send a first request to said server computer;

responsive to said first request, said server computer sending a Locally Administered Address to said workstation computer;

responsive to receiving said Locally Administered Address, said workstation computer being directed by said first program to send a second request;

responsive to said second request, said server computer transmitting an operating system corresponding to said Locally Administered Address to said workstation.

3. A computer readable memory for causing a first computer to present a menu to a plurality of second computers comprising:

a first computer readable storage medium;

a computer program stored in said storage medium;

the storage medium, so configured by said computer program, responsive to a request from at least one second computer, causes the first computer to send a Locally Administered Address to said second computer; and

responsive to a request from said second computer, cause the first computer to transmit a program addressed to said Locally Administered Address to said second computer.

4. A computer implemented process to accomplish booting of a workstation computer from a server computer comprising:

using a first computer, performing the following series of steps:

powering the first computer;

obtaining control of the first computer by means of a first program;

executing, without an operating system, the first program to communicate with a network server;

communicating a first request to a second computer;

receiving a Locally Administered Address from a second computer;

responsive to receiving said Locally Administered Address requesting a boot program;

receiving a boot program corresponding to the Locally Administered Address;

booting the first computer;

using a second computer, performing the following series of steps:

responsive to the first request from the first computer, sending a Locally Administered Address to the first computer; and

responsive to the second request from the first computer, sending a boot program corresponding to the Locally Administered Address to the first computer.

5. A method for administering at a server computer, the booting of a client computer having a Universally Administered Address by assigning a Locally Administered Address to the client computer, the method comprising the computer implemented steps of:

executing instructions from a client computer first memory to load a boot code loader from a client computer second memory, wherein the client computer first memory is a BIOS ROM and the client computer second memory is a nonvolatile, read/write memory:

executing the boot code loader to load a control program from the client computer second memory;

executing the control program to load a set of programs from the client computer second memory without loading an operating system;

executing the set of programs to communicate a first message to a network server;

responsive to said first message, retrieving a Locally Administered Address from the network server;

executing the set of programs to communicate a second message to a network server;

responsive to said second message, receiving at least one program from the network server; and

executing the program at the workstation computer;

whereby the workstation computer is booted from the program.

6. A computer program product on a computer-readable medium for booting a client computer without an operating system by replacing the client computer's Universally Administered Address with a Locally Administered Address, the computer program product comprising:

first instructions from a first memory for loading a set of programs from a second memory, wherein the first memory is a BIOS ROM and the second memory is a nonvolativle, read\write memory;

second instructions for communicating a first request for a Locally Administered Address to a network server;

responsive to receiving said Locally Administered Address, third instructions for communicating a second

request for a second set of programs addressed to said Locally Administered Address; and

responsive to receiving said second set of programs, fourth instructions for initiating execution of the second set of programs;

wherein said second set of programs includes an operating system.

* * * * *